LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84
Il6r
no. 607-612
cop. 2

CENTRAL CIRCULATION AND BOOKSTACKS

The person borrowing this material is responsible for its renewal or return before the **Latest Date** stamped below. **You may be charged a minimum fee of $75.00 for each non-returned or lost item.**

Theft, mutilation, or defacement of library materials can be causes for student disciplinary action. All materials owned by the University of Illinois Library are the property of the State of Illinois and are protected by Article 16B of *Illinois Criminal Law and Procedure.*

TO RENEW, CALL (217) 333-8400.

University of Illinois Library at Urbana-Champaign

APR 2004

When renewing by phone, write new due date below previous due date. L162

FEB 12 1974

Math
IL6N
no. 611
cop. 2

Report No. UIUCDCS-R-73-611

calc2 - A PLATO IV Lesson on Differentiation

by

Axel T. Schreiner

December 1973

THE LIBRARY OF THE
FEB 15 1974
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-73-611

calc2 - A PLATO IV Lesson on Differentiation

by

Axel T. Schreiner

December 1973

Departments of Computer Science and Mathematics
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Digitized by the Internet Archive
in 2013

http://archive.org/details/calc2platoivless611schr

ACKNOWLEDGMENT

The PLATO IV consulting staff, especially Judy Sherwood and Jim Kraatz, helped me many times when TUTOR and I were on different wave lengths. After I was done programming, Rick Blomme and his staff further improved the PLATO system. Professor H. George Friedman, Jr. (as usual) read this paper and removed my hyphens. And Vivian Alsip read my handwriting and produced a perfect manuscript.

Without all this help, I would still be reading in Knuth's book [1] on how to differentiate...

TABLE OF CONTENTS

		Page
1.	Introduction.	1
2.	Formula Management.	2
3.	Formula Representation.	3
4.	The Various Types	5
5.	Input (readin, and postfix)	7
6.	The Choice Page (options)	10
7.	The Differentiation Algorithm (derive).	12
8.	Interfacing with the Student (infix, and output).	18
9.	Wrap up	20
10.	Conclusions	21
REFERENCES		23
A Listing of calc2		24

LIST OF FIGURES

		Page
1.	Storage Layout for the Formulas.	4
2.	Example of a Differentiation Situation	12
3.	Example of the Differentiation Process	13
4.	Layout of an Entry in the Driving Table.	17

LIST OF TABLES

		Page
1.	Types of a Node in the Formula	6
2.	Codes in the Expanded Input String	8
3.	The Input Parser Stack Mechanism	9
4.	The Differentiation Situations	16
5.	Displayed Information in a Differentiation Situation	18
6.	Use of the Student Bank.	22

calc2 - A PLATO IV Lesson on Differentiation

1. Introduction

In the summer of 1971, I wrote a symbolic differentiator as a demonstration program for a course on computer applications in calculus [2]. This program was, in its central part, a direct and unmodified implementation of the algorithm described by Knuth [1], but it was able to differentiate all trigonometric, inverse trigonometric, and hyperbolic functions, as well as e^x, and various logarithms. The program was written in BASIC.

In fall 1972, I decided to recreate this program on the PLATO IV system [3], originally as a demonstration for the various stack-oriented algorithms used during input, differentiation, and output of a formula. BASIC had not been a good language for string- and tree-manipulation (neither is TUTOR, the language of the PLATO IV system), and the recreation was also intended to improve on the implementation of the algorithm.

During the design of this PLATO IV lesson, calc2, however, I decided to make the student a participant rather than a spectator to the differentiation process. calc2 was to control a student's attempts at differentiation by guiding him through the steps and controlling each of his operations. To keep this guidance deterministic from the machine's point of view, the student must differentiate in postorder (i.e., for each operator, he must differentiate the operands before he is allowed to construct the derivative of the phrase involving the operator). An adapting control allows him to dynamically bypass some or all of the simpler steps in differentiation.

Calc2 is table-driven; it only provides the routines to manipulate the data structures which represent the formula and the piece-wise derivatives; the actual differentiation is under control of a table of differentiation algorithms. It was thus possible during the design of this lesson to expand the set of differentiation situations--rules may be easily added or deleted. It is quite

conceivable that a particular table might be introduced (then together with suitably chosen problems!) which only practices the chain rule, etc. Similar effects can be obtained by manipulating the dynamic bypass mechanism.

In this report I intend to describe how calc2 operates. TUTOR, PLATO's authoring language, is ever-expanding, and some coding in calc2 might now be inefficient; however, I believe the basic techniques still are acceptable.

2. Formula Management

Knuth [1] presents a formula to his differentiator as a binary threaded tree. If one intends to perform a lot of (for the mathematician-user necessarily infixed) formula fragment display, and if the implementation language is not suited to tree manipulation, such a tree is not the best data structure.

Instead, I decided to employ one queue and two stacks with variable sized entries.

Initially, the queue is loaded with the given formula in post order (i.e., each operator follows its operands), each element of this queue containing one fragment (operator or operand) of the formula. Both stacks are empty.

As the differentiation progresses, one stack is pushed with phrases of the formula, the other with the derivatives of these phrases; e.g., if a binary operator of the formula is processed, the top two phrases of each stack are combined to form a new top phrase of the derivative stack, and then the top two phrases of the formula stack and the binary operator are combined to form the new top phrase of the formula stack. For display and judging purposes, these processed phrases are maintained with binary operators infixed between their operands and parentheses are employed.

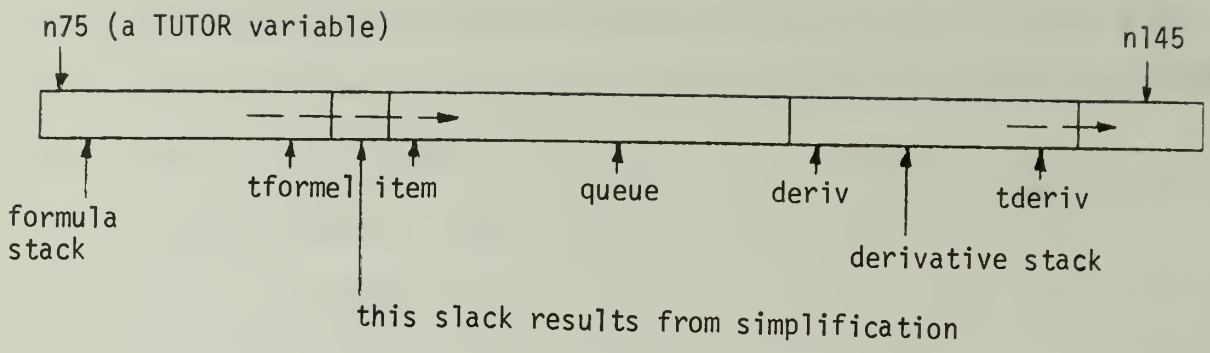
Constructing an infixed formula 'bottom up' allows one to insert the minimal number of parentheses without any effort, provided the actual data

structure underlying the formula is flexible enough to accommodate these parentheses without expanding in size.

3. Formula Representation

PLATO IV operates on a Control Data Cyber 70 computer; the word size is 60 bits, and at the time calc2 was written, TUTOR did not provide for the subdivision of a word. Bit operations (logical and, or, and circular shift) were available, as was a one-dimensional array facility. A student in the PLATO IV system has 150 "student variables" available, and it is the lesson author's responsibility to manage those variables; no dynamic variable allocation is provided in TUTOR.

I decided to allocate the queue and both stacks in one array. The queue occupies the beginning of the array. As elements are disposed of from the beginning of the queue, the formula stack overlays these elements. If we accommodate the parenthesized representation at no increase in space requirements, the queue will not have to be moved. The derivative stack follows the queue in the array. This method is destructive, but once differentiation is complete, formula and derivative exist infixed and either one could then again be converted to post-order to repeat the process.



tformel: pointer to top of formula stack

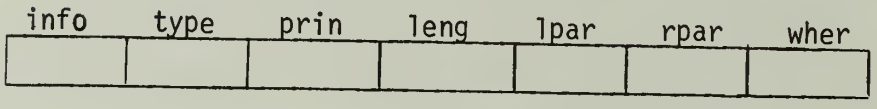
tderiv: pointer to top of derivative stack

deriv: pointer to bottom of derivative stack (one element beyond queue)

item: pointer to current element in queue

Figure 1. Storage Layout for the Formulas

Elements in the queue are one word long, and these "nodes" have the following fields:



info: bits 1...8 : (pointer to) auxiliary information

type: bits 9...14 : type of this element

wher: bits 39...60 : begin position of this element in the master display line

These elements are essentially copied from the queue into the stack, and then the following fields are also used:

rpar: bits 34...38: number of) following this node

lpar: bits 29...33 number of (preceding this node

leng: bits 21...28: length field

poin: bits 15...20: principal type field

The length and principal type field implement the variable sized stack entries and allow simplification: the last node of a phrase contains in its leng field the length of that phrase, i.e., the number of nodes constituting it, and the prin field indicates the principal type of the phrase. This principal type is defined to be the type of the root node of the formula tree into which the phrase could be expanded. If the phrase is a function evaluation, the node representing the function contains in its prin field the principal type of the function argument. It is thus possible to "collapse" the composition of a function and its inverse--like two successive unary minus operators.

4. The Various Types

calc2 allows its user to specify the formula which he wishes to differentiate. The user may use 26 variables a...z, the binary operators +,-, *,/ and ** for exponentiation, the unary minus operator -, and the functions sin, cos, tan, cot, sec, csc (trigonometric), sinh, cosh, tanh, (hyperbolic), asin, acos, atan (inverse trigonometric), and ln, exp, sqrt. Parentheses may be used. Following TUTOR conventions, multiplication takes precedence over division. At most points in calc2, the standard TUTOR formula evaluation mechanisms can be employed; these permit juxtaposition of variable names to denote multiplication, superscript to denote exponentiation, and omission of (most) parentheses in function application; however, in order not to overly complicate the input scanner for the given formula, it must be specified in a more rigorous, conventional syntax; we assign the unary minus the highest precedence so that the phrase a**-b would be legitimate. Upon input we have two types of terminal operands: numerical values which for simplification purposes are assigned the types 0, 1, and v, and variables which are assigned the type c. Only during differentiation do we make the further distinction between type c for constant variables and type x for the differentiation variable.

type	type name	info
1	x(variable)	pointer to variable storage at random value for x
2	c(constant)	pointer to variable storage at random value
3	0 (zero)	pointer to constant storage at value 0
4	1 (one)	pointer to constant storage at value 1
5	v (value)	pointer to constant storage at value
6	+	
7	-	
8	*	
9	/	
10	**	
11	-(unary)	
12	ln	
13	exp	
14	sin	
15	asin	
16	cos	
17	acos	
18	tan	
19	atan	
20	cot	
21	sec	
22	csc	
23	sinh	
24	cosh	
25	tanh	
26	sqrt	

Table 1. Types of a Node in the Formula
(to be constructed by the input-parser)

In TUTOR, characters a through z have the internal codes 1 through 26; we allocate variable storage (used for randomization during answer judgment) into variables v1 through v26 and thus have the info fields of types x and c also store the character associated with the node.

Values pose a bigger problem. When judging phrase answers I randomize all variables (types c and x), evaluate the correct phrase known to the lesson and the answer phrase given by the student, and compare. In order to evaluate efficiently, I intended to use TUTOR's "compute" command which takes a character string, compiles it (it must be a formula), and evaluates it. Such a formula may contain the usual operators, numbers, and "defined names". Names are defined with a "define" statement which assigns at compile time either a variable location or a constant as a string replacement for the name. Treating the variables a...z is simple: they are defined to be variable locations v1 through v26, and before using compute, random numbers can be placed into those locations. How should I maintain values? There exists, of course, a table of values, and the info field of types 0, 1, and v, points to an entry in this table. But TUTOR does not provide a conversion from floating point to character string in memory. I could have stored values as character strings, but that would have severely limited the simplification techniques which may introduce new values. I, therefore, "defined" the names A, B, ... for the locations of the value table, and in phrases presented to the "compute" command, these names are used instead of values. This meant then that output of these phrases to the screen had to be accomplished interpretively: strings between values could be shown directly, values had to replace their names in the phrase strings.

5. Input (readin, and postfix)

The input parser accepts a formula and constructs the queue from it;

the formula is in infix notation and in a conventional syntax, the queue is in postorder, parentheses-free, and encoded into nodes. The value table must also be constructed. The input parser has two phases: first the formula string is converted from a character string into a sequence of codes, one per computer word. Blanks are eliminated and illegal characters are rejected; function names including left parentheses are verified. The codes are outlined in Table 2. They overlay the area which is then occupied by the (more compact) queue.

Code	Meaning
1 ... 26	variable a ... z
27	+
28	-
29	*
30	/
31	**
32	the next computer word contains a floating-point value
33	the next computer word contains a subcode (this always indicates a left parenthesis)
34	)

subcode	meaning
1	(
2 ... 16	functions of type = subcode + 10

Table 2. Codes in the Expanded Input String

The second phase of the input parser is a syntax analyzer, driven by an operator precedence grammar. A stack is used and the coded formula is converted into the

postorder nodes in the queue. This part of the parser is straightforward and I therefore just present it as Table 3.

state	next symbol	code	action	next state
expect operand	variable	1 ... 26	emit	expect operator
	value	32	emit and enter into value table	expect operator
	function or left paren	33, subcode	stack, increase # of parens	expect operand
	+	27	ignore unary plus	expect operand
	-	28	stack or collapse unary minus	expect operand
	other		SYNTAX error	retrieve new formula
expect operator	+, -, *, /, **	27 ... 31	emit from stack and push by precedence	expect operand
	)	34	emit from stack decrease # of parens	expect operator
	end of formula		if no parens open	exit
	other		SYNTAX ERRØR	retrieve new formula

Table 3. The Input Parser Stack Mechanism

TUTOR is not overly cooperative in constructing either phase of the input scanner: the formula is read as a student response to a TUTOR arrow, and at least conversion to the first internal representation should have been possible by using TUTOR's rather eye-pleasing array of answer-judging techniques. Unfortunately, no coding could be devised to overcome certain time slicing problems

which still exist: answer judging may not take more than 25ms of computer time or the user will be swapped and some of the judging material is lost. I was unable to determine whether my code was troubled by bugs in TUTOR or whether I was violating a principle of the language. Consequently, the first phase is accomplished by "brute force", the characters of the given formula are spread out, one to a computer word, and then analyzed. The coding is certainly inefficient, FORTRAN-style, and TUTOR's limits on the size of a program segment ("unit") were felt to be a severe restriction.

The second phase of the scanner charged with creating postorder is simple, except for the treatment of the wher field in each output node: eventually our lesson will present on each page a differentiation situation taken from the given formula. This formula is also shown on each page, and an arrow indicates the location of the presented situation in context. We thus must maintain with each node an indication where it is displayed in the formula. This information is maintained in the wher field, and it must be stacked for operators during the postorder conversion. The information is obtained by writing the element onto the screen (with a standard-length format for constants) and observing how much space is needed for writing. (Theoretically, TUTOR's screen location variable "where" could be used, but "where" had not yet been completely debugged.)

In order to generate the given formula display efficiently on each new page (since a selective screen erase conserving the formula turned out to be too slow), it is generated and saved in infixed form in memory, with values replaced by capital letter-defined names, while the endorder queue is constructed.

6. The Choice Page (options)

Having successfully presented a formula to the system, the user is

given four choices: he may read a new formula (implemented as a return to the reinitialized input scanner), replace a variable by a value, evaluate or differentiate the given formula.

The replacement is implemented as a simple substitution: an entry in the value table is made and in all nodes referencing the variable, the type and info fields are changed to indicate a value and to point to the entry in the value-table. All wher fields to the right of the value must be updated to allow for the display of the value, and in the infix representation of the formula, the letter variable must be replaced by a capital letter value reference. (Since in TUTOR such a capital letter takes two six-bit codes, the first of which indicates capitalization, the replacement would be rather cumbersome. Instead, each letter variable in infix is preceded by a blank character, and this blank-code then can be overwritten with a shift-code during replacement.)

Evaluation is implemented by presenting the infix string to the compute command. This, of course, must be preceded by assigning user-supplied values to the variables in the given formulas; these values are stored into those locations in the student bank which the defined names for the variables reference.

The differentiation driver requests the name of the differentiation variable and changes the type field of all nodes referencing this variable to indicate type "x". Control then passes to the differentiation driver. A special, non-published choice is also available; differentiation can be requested to be automatic, bypassing any questioning of the user. This is accomplished by setting the request control for the questioning to a special value as will be outlined below.

7. The Differentiation Algorithm (derive)

Consider the differentiation situation in Figure 2:

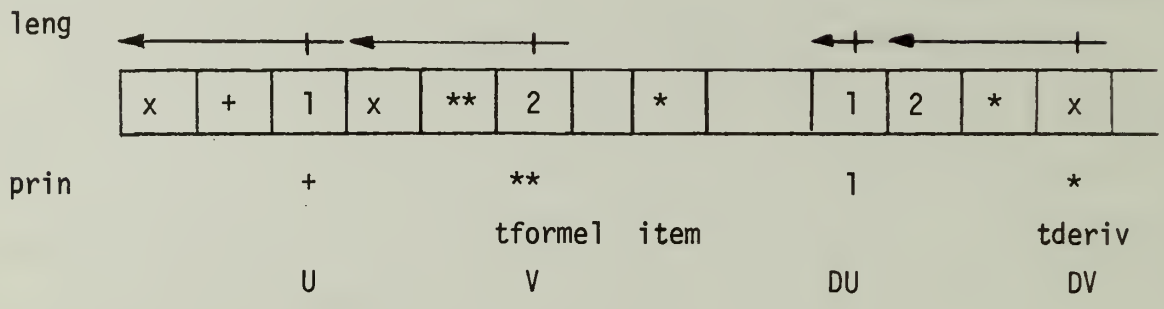


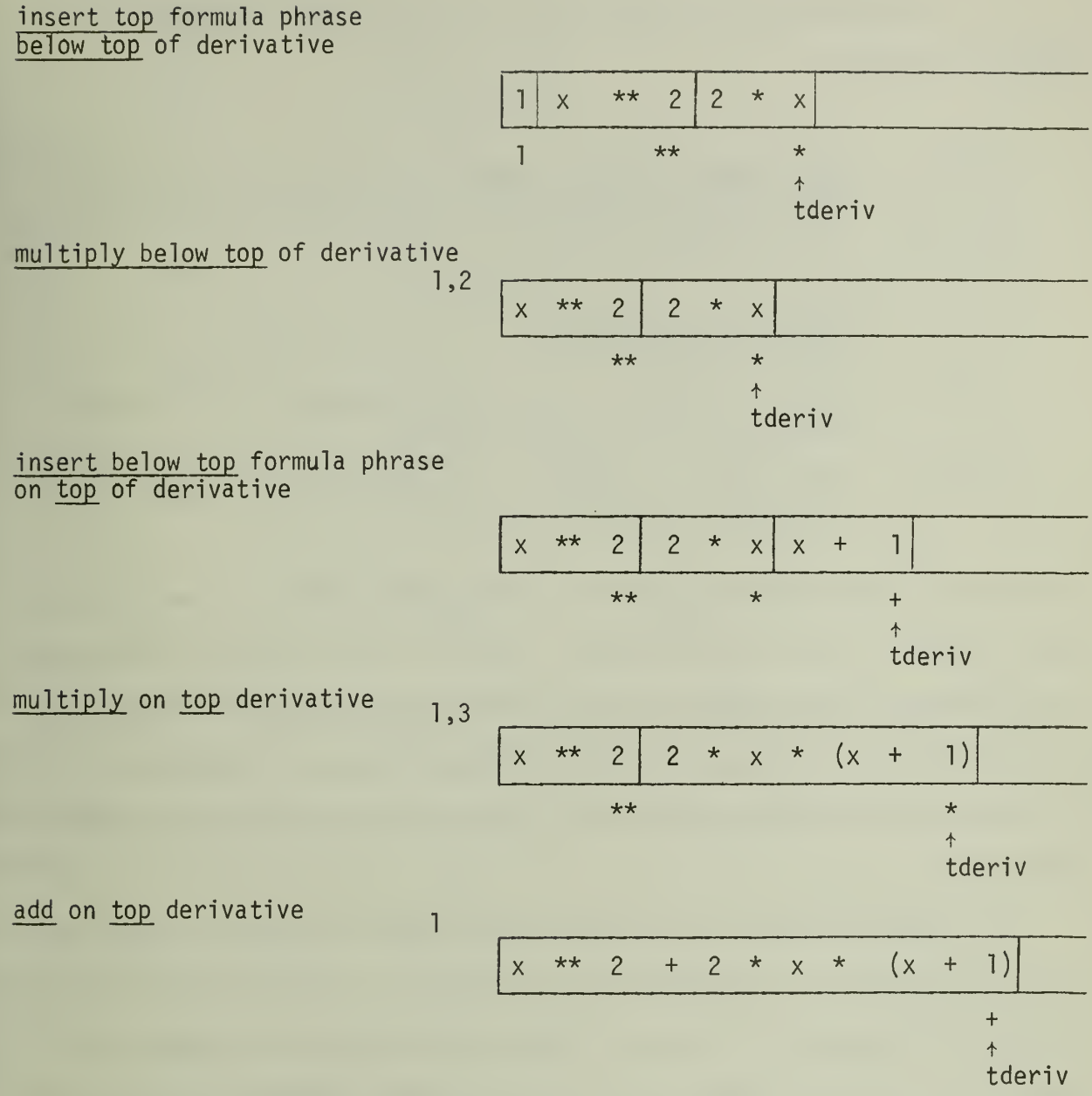
Figure 2. Example of a Differentiation Situation

item selects a multiplication operator, the corresponding operands are (x+1) and (x**2), both have non-zero derivatives (assuming a differentiation with respect to x), the corresponding differentiation rule is the product rule

$$D_X(U*V) = DU*V+U*DV \ .$$

With the aid of the leng fields, the phrases U,V,DU,DV can be located from the top of the respective stacks; the relevant pointers have been added to the figure.

The 'algorithm' described in Figure 3 will produce the correct derivative (the figure shows the stepwise results in the derivative stack only). To complete processing the item '*', we now have to infix it into the formula stack. This, however, can be accomplished by the same routines which perform multiplication in the derivative stack (an overflow of the formula stack is impossible).



1 any operation is done by infixing the operator between the relevant phrases.

2 a multiplicative factor of '1' can be discarded.

3 since the precedence of the second phrase (principal type '+') is lower than the precedence of the operator to be infixed, '*', we add parentheses to the second phrase.

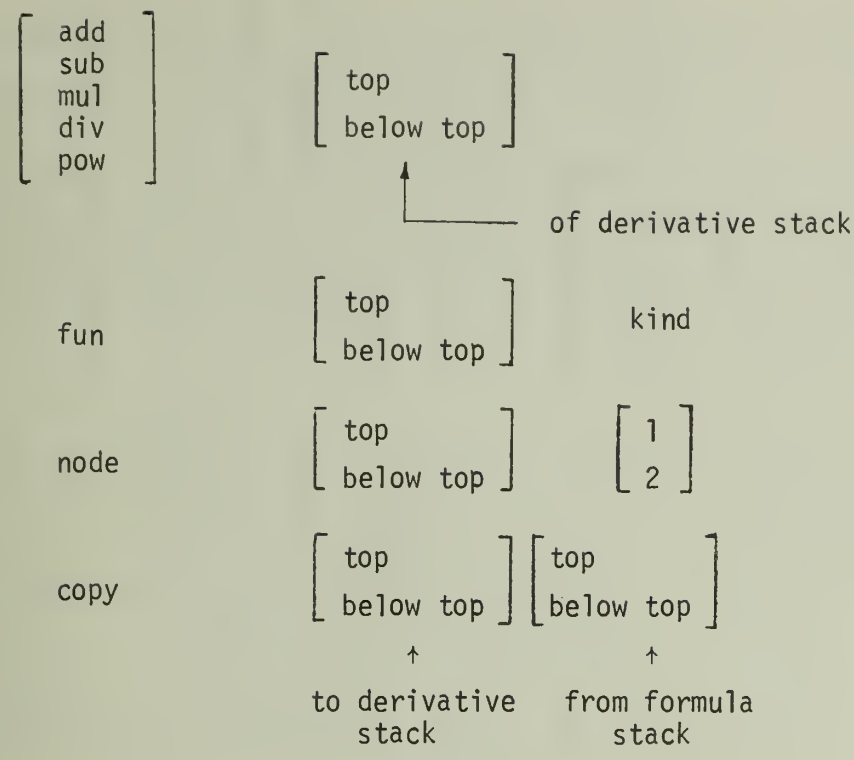
Figure 3. Example of the Differentiation Process

The differentiation machine is built from eight processors for machine operations:

add, sub, mul, div, pow	perform the indicated binary operation to the indicated stack element and the stack element below as right and left operand respectively.
fun	perform the indicated function (unary operator) on the indicated stack element.
copy	copy the indicated formula stack element to the indicated position in the derivative stack.
create	create the indicated node into the indicated position in the derivative stack.

The first six processors can operate on either the function or the derivative stack--they don't know the difference. Each will first attempt an appropriate simplification (discard additive zero, multiplicative or power one, evaluate any operation on two constants, collapse inverse function composition) and if it fails, will infix the operator and correct the result by parenthesization if necessary.

These processors are slightly more generalized than is needed for their application in the differentiation machine. This differentiation machine therefore is implemented with a restricted machine language to access the processors: (it should become clear below, why this machine only seems to operate on the derivative stack.)



The machine selects (by inspection of the item node and DU and DV, whichever is appropriate) one of 28 differentiation situations (compare the table below). The machine's driving table then contains a differentiation algorithm, written in 'machine language', for each differentiation situation.* This algorithm is executed by driving the processors to produce the derivate phrase corresponding to item. Finally, the node item is passed to one of the processors to update the formula stack, item is advanced to the next node in the queue, and the algorithm repeats.

---

*Actually, the first two situations 'constant' and 'variable' are recognized by the machine as special cases and are not handled through the table--it seemed silly to add a special machine instruction for these two cases.

situation number	prin DU	prin DV	item	implemented differentiation rule [c --- constant; U,V --- dependent functions]	
1	--	--	differentiation variable	$Dx = 1$	variable
2	--	--	constant	$Dc = 0$	constant
3	any	any	+	$D(U+V) = DU + DV$	sum
4	any	any	-	$D(U-V) = DU - DV$	difference
5	0 any	any 0	*	$D(c*U) = c\ DU$	factoring out a constant
6	any,≠0	any,≠0	*	$D(U*V) = DU\ V - U\ DV$	product rule
7	0	any	/	$D(c/V) = -c\ DV/V^2$	chain rule, factoring a constant
8	any,≠0	0	/	$D(U/c) = DU/c$	factoring a constant
9	any,≠0	any,≠0	/	$D(U/V) = (DU\ V - U\ DV)/V^2$	quotient rule
10	any	0	**	$D(U^c) = c\ DU\ U^{c-1}$	chain rule, constant exponent
11	0	any,≠0	**	$D(c^V) = DV\ c^V\ \ln(c)$	chain rule, constant base
12	any,≠0	any,≠0	**	$D(U^V) = DU\ V\ U^{V-1} + DV\ U^V\ \ln(U)$	chain rule
13	any	--	unary -	$D(-U) = -DU$	factoring out a constant
14 ... 28	any	--	function	$D(f(U)) = D_U f\ DU$	chain rule

Table 4. The Differentiation Situations

The driving table is currently somewhat wastefully implemented: it takes up 550 words of common storage (which is about 8% of the entire lesson) and is read-only. The table head contains a list of 28 words pointing to the beginning of the table entry for the respective differentiation type. An entry has the format outlined in Figure 4.

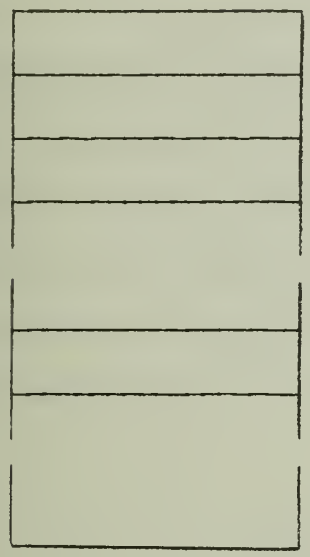

code for display (see below)
# of words used to store structure formula
# of characters in structure formula
structure formula
# of commands to follow
commands for this differentiation rule

Figure 4. Layout of an Entry in the Driving Table

A command consists of two or three words containing the command type and the application information; this could be very much compressed, but at the time the above implementation was considered to be the most efficient in terms of speed.

If, for example, one would like to practice only with polynomials, the driver routine could be modified to recognize fewer types, and a table could be substituted to analyze only, e.g., x^n rather than $f(x)^n$ (where the latter requires the chain rule in the structure formula). This is where the real flexibility of this approach lies. Of course, in this case one would have to restrict the original formula inputs, e.g., to preloaded exercises.

Alternatively, the driver and table can be expanded to make further cases, distinguishing, e.g., sin(x) and sin(f(x)) as different differentiation situations. This expansion was actually (and without any problem) performed while the lesson was written, to have 2 multiplication, and 3 division, and exponentiation situations.

8. Interfacing with the Student (infix, and output)

The algorithm described above actually operates in the background of the lesson. In each differentiation situation, it is used to generate internally the correct (although probably incompletely simplified) solution. The student is then provided with the environment of the situation (generated from the infixed phrases in the stacks before differentiation) and must answer some questions. If he is unable or unwilling to answer, hints or the complete answer can be given from the differentiation algorithm.

situation	environment
x or constant	U = x or constant DU = ? recognized and handled as a special case
unary operator	U = argument DU = derivative of argument question for structure question for this solution
binary operator	U,V = arguments DU,DV = their derivatives question for structure question for this solution

Table 5. Displayed Information in a Differentiation Situation

How many phrases constitute the environment is controlled by the display code, the first entry in the command table (this proved to be an efficient implementation; it is not strictly a necessary table entry). The correct

structure formula (e.g., in situation 6 it would be D(U*V) = DU*V + U*DV, and commutativity must be observed) is available from the command table. The correct solution in the individual case is available from the differentiation algorithm.

The student is first questioned for the correct structure formula. His answer is to involve only the quantities c (for a constant), U, V, DU, and DV (whichever is appropriate). The answer is judged by assigning random values to c ... DV, and evaluating the student answer and the system-known formula. If the two do not match, in turn DU and DV (if appropriate) are set equal to one to test by reevaluation whether the student may have forgotten DU or DV. Finally, DU and DV are set equal to zero to test for a missing term in the structure formula. If no guess matches, the student is invited to key -help- to see the correct answer (if he does not wish to retry his response).

Once a correct structure is established, the student must substitute the values at hand. This algebraic answer is tested with all randomized variables, and if it does not match the known correct answer, -help- is available to see this correct answer.

The above seemed to be a reasonable interface: because of the -help- facility the student cannot get stuck, and his progress could be accounted for. The algorithm is completely independent of the presented situation and is relatively efficient in execution speed. Suggested improvements include:

1) count the number of operators in the formulas to be matched to force a minimal (reasonable) simplification on the student;

2) if the student's answer is more simplified than the system's, substitute;

3) supply 'wrong' answers in the command table, and aid-information with them. This, however, can only improve the response on the structure formula and the implemented correction method seems already very satisfactory;

4) there should be a way to check on the student's algebra in substitution ... .

1) is a minor extension (thanks to TUTOR) and as such already planned. 2) is fairly prohibitive due to the scanning problems: such a formula then would have to follow the input parser's syntax rather than the far more lenient TUTOR syntax in order to be readily converted into the stack. Also, execution of the scanner is not fast enough to allow this as a rather frequent operation.

There is one more control mechanism to be described: with each differentiation situation a counter is associated. These counters can be used to prohibit the situation (it may be unknown to the student), or to bypass questioning. Currently, questioning is bypassed once the situation has been answered correctly one or more times, depending on the situation. With a suitable control strategy, rather elaborate learning paths could be established, leading (within prescribed examples) from simple to more complex differentiation situations, etc. The counters very definitely eliminate boring repetition.

9. Wrap up

Once the differentiation is complete, the user can again evaluate formula and derivative for various arguments, and he can retain either one for another round in differentiation. Formulas of medium complexity can undergo about three differentiations before the available stack and queue space is exhausted. The formulas are retained by expansion into a form that is acceptable to the second phase of the input parser. Formula evaluation is accomplished by the same routine as described in section 6.

10. Conclusions

PLATO IV and TUTOR do allow for inefficient but rather complex algebraic symbol manipulation. The presented processors, representations, and techniques are reasonably compact and versatile in application. The table implementation of the differentiator should be easily adaptable to a variety of levels of experience and problems for students--it is at least being contemplated to have a calculus exercise system on PLATO IV in which calc2 might play a large role. Currently under investigation is the implementation of a similarly table driven integration teacher.

variable number	first usage	second usage	third usage
1 . . . 26	random values for the variables a ... z in algebra-answer judging	random values for the variables U,V,c, DU,DV in structure-answer judging --- storage of the character form of the structure answer	scratch locations for subroutines, and some display controls
27	deriv		
28	cpt(top of constant table)		
29	tderiv		
30	tformel		
31	dopt (operator to be differentiated)		
48 . . . 56	stack for postorder conversion infix form of formula undergoing differentiation	semi-global variables for differentiation: e.g., item typDU typDV infix form of algebra-answer, and argument formulas for display	
57 . . . 74	constant table 0 1 2 A --- Ø 15 values		
75 ... 145	node lists (see Figure 1)		
146	algebra help request counter		
147	structure help request counter		
148 . : 150	question control (counters for each differentiation situation, and their restore values)		

Table 6. Use of the Student Bank

REFERENCES

[1] D. Knuth, "The Art of Computer Programming," Vol. 1, Section 2.3.2, 1968, Addison Wesley, Reading, Mass.

[2] A. Schreiner, "Computer Calculus," Chapter 7, 1972, Stipes, Champaign.

[3] D. L. Bitzer, B. Sherwood, and B. Tenczar, Computer-based Science Education, CERL Report X-37, Dec. 1972 (Computer-based Education Research Laboratory, University of Illinois at Urbana-Champaign

A Listing of calc2

```
                                                                                                                    TITLE
*LIST SYMBOLS
*
*       DIFFERENTIATION TUTORIAL
*
*       ATS 72/73
*
RESTART MYSELF

                                                                                                                    MYSELF
UNIT    MYSELF          $$ INITIALIZE QUESTION CONTROL
TERM    RESET
DEFINE  REQUEST,REQUEST=148
        HELPA=#146 $$ HELP WITH ARITHMETIC;
        HELPS=#147 $$ ↑, STRUCTURES
CALC    N(REQUEST)→0010111111112312112312
        N(REQUEST+1)→0122311112121223122312
        N(REQUEST+2)→0231212121212121127777
        HELPA→↑FLPS→0
*       N(REQUEST,,) HOLDS 6 BITS/DIFFERENTIATION TYPE
*       BIT 123 HOLD RESET VALUE, 456 COUNTING VALUE
*       IF CORRECT ANSWER, COUNTING VALUE IS DECREASED
*       7 --- INHIBITED

                                                                                                                    CALC2
ENTRY   CALC2 +$ SETUP AND DUN↑7T CHANGE QUESTION
*
*       PRACTICING DIFFERENTIATION
*       CALCULUS TUTORIAL
*
*       ATS 5/7/73
*
TERM    CALC?
AT      3125
WRITE   ↑LOADING THE ALPHABET ---
CHARSET UICALC.CHARSET
ALTFONT ALT
ERASE
RESTART CALC?
COMMON  CALC2,DERIVATIVE,550,READ ONLY
JOIN    DATELINE
AT      1010
SIZE    2
WRITE   CALCULUS
SIZE    0
AT      1310
WRITE   [2] PRACTICING DIFFERENTIATION
AT      1510
WRITE   ¬↑M¬↑N¬↑0 5/23/73
PAUSE   3
AT      2810
WRITE   REMEMBER↑!
        -↑N↑E↑X↑T- WILL MOVE YOU ALONG.
        ↑IF NOTHING HAPPENS, HIT -↑N↑E↑X↑T-
NEXT    READIN $$ TO AVOID RESETTING QUESTION CONTROL
*
```

```
53  *
54  *                                                                    DATELINE
55  UNIT    DATELINE
56  *
57  *       DISPLAY LESSON AND DATE IN LINE 1
58  *
59  DEFINE  DATES
60          DATE1=¬A,DATE2=¬9          $$ SCRATCH
61  MODE    REWRITE
62  AT      101
63  DATE    DATE1
64  CLOCK   DATE2
65  SHOWA   DATE1,10
66  WRITE     ↑C↑A↑L↑C↑U↑L↑U↑S 2
67  WRITE     PRACTICING DIFFERENTIATION
68  AT      155
69  SHOWA   DATE2,10
70  MODE    WRITE
71  *
72  *
73  *
-------------------------------------------------------------------------- DEFINE1
                                                                       ERROR
75  UNIT    ERROR
76  DEFINE  STUDENT
77          A=V1,B=V2,C=V3,D=V4,E=V5,F=V6,G=V7,H=V8
78          I=V9,J=V10,K=V11,L=V12,M=V13,N=V14,O=V15,P=V16
79          Q=V17,R=V18,S=V19,T=V20,U=V21,V=V22,W=V23,X=V24
80          Y=V25,Z=V26 $$ DURING JUDGING USED FOR RANDOMS
81          ↑A=V60,↑B=V61,↑C=V62,↑D=V63,↑E=V64,↑F=V65,↑G=V66
82          ↑H=V67,↑I=V68,↑J=V69,↑K=V70,↑L=V71,↑M=V72,↑N=V73
83          ↑O=V74 $$ NAMED CONSTANTS
84          ↑U=V1,↑V=V2,↑D↑U=V4,↑D↑V=V5 $$ FOR STRUCTURE CHECK
85          TAN(ZZ)=SIN(ZZ)/COS(ZZ),COT(ZZ)=COS(ZZ)/SIN(ZZ)
86          SEC(ZZ)=1/COS(ZZ),CSC(ZZ)=1/SIN(ZZ)
87          ASIN(ZZ)=ARCTAN(ZZ/SQRT(1-ZZ↑2))
88          ACOS(ZZ)=¬P/2-ARCTAN(ZZ/SQRT(1-ZZ↑2))
89          ATAN(ZZ)=ARCTAN(ZZ),SINH(ZZ)=(EXP(ZZ)-EXP(-ZZ))/2
90          COSH(ZZ)=(EXP(ZZ)+EXP(-ZZ))/2
91          TANH(ZZ)=(EXP(ZZ)-EXP(-ZZ))/(EXP(ZZ)+EXP(-ZZ))
92          ¬M=EXP(1),¬A=¬P $$ INO CONSTANTS
93  DEFINE  MINE,STUDENT
94  ***  NODE LAYOUT
95          IINFO(ZZ)=ZZ$CLS$51      $$ BIT  1- 8  ¬↑D INFO
96          ITYPE(ZZ)=ZZ$CLS$45      $$ BIT  9-14  TYPE
97          IPRIN(ZZ)=ZZ$CLS$39      $$ BIT 15-20  PRINCIPAL TYPE
98          ILENG(ZZ)=ZZ$CLS$31      $$ BIT 21-28  LENGTH SUBTREE
99          ILPAR(ZZ)=ZZ$CLS$26      $$ BIT 29-33  ¬$ ( PRECEEDING
100         IRPAR(ZZ)=ZZ$CLS$21      $$ BIT 34-38  ¬$ ) FOLLOWING
101         IWHER(ZZ)=ZZ$MASK$07777777 $$ REST   DISPLAY LINE
102         ZINFO=0400777777777777777777
103         ZTYPE=0777007777777777777777
```

```
106          ZLPAR=0777777777760377777777
107          ZRPAR=0777777777777407777777
108          ZWHER=0777777777777770000000 $$ MASKS FOR ITEMS
109          INFO(ZZ)=N(ZZ)$CLS$ 9$MASK$255
110          TYPE(ZZ)=N(ZZ)$CLS$15$MASK$63
111          PRIN(ZZ)=N(ZZ)$CLS$21$MASK$63
112          LENG(ZZ)=N(ZZ)$CLS$29$MASK$255
113          LPAR(ZZ)=N(ZZ)$CLS$34$MASK$31
114          RPAR(ZZ)=N(ZZ)$CLS$39$MASK$31
115          WHER(ZZ)=N(ZZ)$MASK$000000000000007777777 $$ RETRIEVE ITEMS
116     *** TYPE INFORMATION IN NODES
117          TYPEX=1    $$ TYPE ↑A↑7 (INFO TO CHARACTERS)
118          TYPEC=2    $$ TYPE ↑C↑7 (INFO TO CHARACTERS)
119          TYPE0=3    $$ TYPE ↑0↑7 (INFO TO VALUE)
120          TYPE1=4    $$ TYPE ↑1↑7 (INFO TO VALUE)
121          TYPEV=5    $$ TYPE ↑C↑7 (INFO TO VALUE)
122          TYPEADD=6  $$ TYPE ↑7+↑7
123          TYPESUB=7  $$ TYPE ↑7-↑7
124          TYPEMUL=8  $$ TYPE ↑7*↑7
125          TYPEDIV=9  $$ TYPE ↑7=↑7
126          TYPEPOW=10$$ TYPE ↑7¬↑W↑7
127          TYPEMIN=11$$ TYPE ↑7- (UNARY)↑7
128          TYPELOG=12$$ TYPE ↑7LN↑7
129          TYPEEXP=13$$ TYPE ↑7EXP↑7
130          TYPESIN=14$$ TYPE ↑7SIN↑7
131          TYPEASN=15$$ TYPE ↑7ASIN↑7
132          TYPECOS=16$$ TYPE ↑7COS↑7
133          TYPEACS=17$$ TYPE ↑7ACOS↑7
134          TYPETAN=18$$ TYPE ↑7TAN↑7
135          TYPEATN=19$$ TYPE ↑7ATAN↑7
136          TYPECOT=20$$ TYPE ↑7COT↑7
137          TYPESEC=21$$ TYPE ↑7SEC↑7

-------------------------------------------------------------------------------- DEFINE2
139          TYPECSC=22$$ TYPE ↑7CSC↑7
140          TYPEHSN=23$$ TYPE ↑7SINH↑7
141          TYPEHCS=24$$ TYPE ↑7COSH↑7
142          TYPEHTN=25$$ TYPE ↑7TANH↑7
143          TYPESQR=26$$ TYPE ↑7SQRT↑7
144          VALUE(ZZ)=PRIN(ZZ)=TYPE0
145     *** CORE LAYOUT
146          FORMULA=N48      $$INFIX OF FORMEL FOR DISPLAY
147          FORLOC=48        $$LOCATION OF THIS INFIX
148          FORMAX=8         $$SIZE IN WORDS
149          SHIFT=070        $$SHIFT CODE
150          FORMLEN=6        $$ LENGTH OF FORMAT
151          FORSIZE=N56      $$ HOLDS LENGTH OF FORMULA IN CHAR
152          TABLE=57         $$ ¬↑U BOTTOM OF CONSTANTTABLE 0,1,2,..
153          START=75         $$ ¬↑U BEGIN FROMULA
154          DERIV=N27        $$ ¬↑U BEGIN DERIVATIVE, END FORMULA
155          END=145          $$ ¬↑U END DERIVATIVE
156          ITEM=N36         $$ ¬↑U CURRENT ATOM IN FORMULA
157          TFORMEL=N30      $$ ¬↑U TOP IN FORMEL
158          TDERIV=N29       $$ ¬↑U TOP IN DERIVATIVE
```

```
159           MPT=N10          $$ > BY↑: DMOVEUP, DMOVEDO
160           MSI=N11          $$ ¬↑W
161           APT=N12          $$ > BY↑: ADD SUB MUL DIV POW UNARY
162           WHE=N19          $$ ¬↑W
163           BPT=N13          $$ ¬↑W
164           LPT=N17          $$ > BY↑: LOOKC
165           VAL=V20          $$ ¬↑W
166           LJUNK=N18        $$ ¬↑W
167           CPT=N23          $$ ¬↑W TOP VALUE
168           FROM=N20,INS=N21 $$ > BY↑: INSERT,NODE
169           OPR=N22
170           MO1=N1           $$ LOCAL SCRATCH
171           MO2=N2,MO3=N3,MO4=N4,MO5=N5,MO6=N6,MO7=N7
172           MO2LOC=2         $$ LOCATION OF MO2
173           MO1LOC=1
174           XPT=N14          $$ > BY↑: INFIX
175           XSI=N15,XUS=N16
176           PAR1=N32         $$ SCRATCH DURING EVALUATION
177           LOPR=N24         $$ TO STACK OPERATOR >PFSTACK
178           STACK=32         $$ BEGIN OF POSTFIX STACK
179           STOP=N23         $$ ¬↑D TOP OF POSTFIX STACK
180           LPARCT=N25       $$ COUNT (
181           DVAR=N35         $$ NAME OF DIFF VARIABLE
182           HELPA=N146        $$ ARITHMETIC HELP CALLED
183           HELPS=N147        $$ STUCTURE HELP CALLED
184           COPT=N34         $$ POINTER TO ENTRY IN COMMON
185           WORKFOR=N39      $$ WORKING FORMULA STO
186           WORKSIZ=N47      $$ SIZE ¬↑W IN CHAR
187           WORKMAX=8
188           WORKLOC=39
189           TYP↑D↑U=N37       $$ PRINCIPAL TYPE OF ↑D↑U
190           TYP↑D↑V=N38       $$ PRINCIPAL TYPE OF ↑D↑V
191           PAR2=N33         $$ SCRATCH IN DERUNA
192           DOPT=N31
193           REQUEST=148      $$ QUESTION CONTROL
194           ↑DVAL=V6          $$ RANDOMIZED STUDENT ANSWER
195           ↑DFORM=N7         $$ ALPHA STUDENT ANSWER
196           ↑DSIZE=N15        $$ LENGTH OF ¬↑W
197           ↑DPT=N16          $$ COMPUTED CODE POINTER
198           ↑SPT=N17          $$ COMPUTED CODE POINTER
199           INLIST=NC(NC55O)   $$ LIST OF CHARACTERS FOR INFIX
200 CALC      MO1→REQUEST      $$ RESETTING QUESTION CONTROL
201           BRANCH (N(MO1+2)$MASK$O70)=O70,O,X $$ NOT YET QUESTIONED
202 1LOOP     N(MO1)→N(MO1)$MASK$O7070707070707070707O
203           N(MO1)→N(MO1)$UNION$(N(MO1)$CLS$57)
204           BRANCH (MO1→MO1+1)¬<REQUEST+2,1LOOP,X
205 JUMP      READIN           $$  AND LOAD AN INFIX FORMULA

                                                                                DBG INDEX
                                                                                    INDEX
207 UNIT      INDEX
208 TERM      INDEX
209 INHIBIT ERASE
210 NEXT      INDEX
```

```
213  WRITE   ↑TRANSFERRING TO THE ↑C↑A↑L↑C↑U↑L↑U↑S INDEX
214  ERASE   13
215  LESSIN  ↑7UICALC↑7
216  GOTO    ERROR %AND% (MALLOT-MUSE<2000), TOOMUCH, X
217  JUMPOUT ↑7UICALC↑7.↑7INDEX↑7
218  GOTO    TOOMUCH
219  *
                                                                    NOTE
220  UNIT    NOTE
221  TERM    NOTE
222  NEXT    NOTE
223  INHIBIT ERASE
224  AT      2810
225  MOVE    REWRITE
226  WRITE   ↑TRANSFERRING TO THE NOTE FACILITY
227  ERASE   15
228  LESSIN  ↑7CALCNOTE↑7
229  GOTO    ERROR %AND% (MALLOT-MUSE<2000), TOOMUCH, X
230  JUMPOUT ↑7CALCNOTE↑7.↑7NOTE↑7
231  GOTO    TOOMUCH
232  *
                                                                    DEBUG
233  UNIT    DEBUG
234  TERM    DEBUG
235  LAB     ERROR
236  BASE
237  DATA    DEBUG1
238  ERASE
239  AT      1110
240  WRITE   ↑WE WILL COMPUTE AND DISPLAY THE RESULT IN ALL CODES
241  AT      3210
242  WRITE   ↑N↑E↑X↑T AGAIN, ↑L↑A↑B NEW FORMULA, ↑D↑A↑T↑A POSTFIX
243  ARROW   1010
244  STORE   VC551
245  OK
246  AT      2010
247  WRITE   INT
248          REAL
249          OCT
250          ALP
251  AT      2016
252  SHOW    VC551
253  AT      2116
254  SHOWS   VC551
255  AT      2216
256  SHOWO   NC551
257  AT      2316
258  SHOWA   NC551
259  NEXT    DEBUG
260  *
261  *
262  *
                                                                    DEBUG1  29
263  UNIT    DEBUG1
```

```
ERASE
RASE
NEAT    DEBUG
DATA    DEBUG
LAB     ERROR
AT      201
WRITE   -N- -INFO- -TYPE- -PRIN- -LENG- -LPAR- -RPAR- -WHER-
AT      3201
WRITE   MO1=
SHOW    N1
WRITE   --
SHOWA   N1
WRITE   --
SHOWO   N1
WRITE   --
SHOWS   V1
AT      1055
WRITE   TFORMEL=
AT      1155
SHOW    TFORMEL,3
AT      1355
WRITE   DERIV=
AT      1455
SHOW    DERIV,3
AT      1655
WRITE   ITEM=
AT      1755
SHOW    ITEM,3
AT      2555
WRITE   ↑N↑E↑X↑T ↑D↑A↑T↑A
        COMPUTE

        ↑L↑A↑B
        FORMULA
DO      DEBUG2,MO1,START,TDERIV
*
*
*
                                                                DEBUG2
UNIT    DEBUG2          $$ UNTANGLE A NODE
AT      (MO1-START+4)≠100+1
SHOW    MO1,3
AT      (MO1-START+4)≠100+5
JOIN    TYPE(MO1)-3,DEBUGC,DEBUGV,DEBUGV,DEBUGV,X
AT      (MO1-START+4)≠100+12
WRITEC  TYPE(MO1)-1,00,X,C,0,1,V,+,-,≠,Ξ,*,-↑U,LN,EXP,
        SIN,ASIN,COS,ACOS,TAN,ATAN,COT,SEC,CSC,SINH,COSH,
        TANH,SQRT,>>>,
SHOW    TYPE(MO1),3
AT      (MO1-START+4)≠100+20
WRITEC  PRIN(MO1)-1,00,X,C,0,1,V,+,-,≠,Ξ,*,-↑U,LN,EXP,
        SIN,ASIN,COS,ACOS,TAN,ATAN,COT,SEC,CSC,SINH,COSH,
        TANH,SQRT,>>>,
SHOW    PRIN(MO1),3
```

```
320   SHOW    LPAR(MO1)•3
321   AT      (MO1-START•4)≠100•42
322   SHOW    RPAR(MO1)•3
323   AT      (MO1-START•4)≠100•49
324   SHOW    WHER(MO1)•3
                                                                                DEBUGV
325   UNIT    DEBUGV
326   SHOW    V(INFO(MO1)•TABLE)•3•2
                                                                                DEBUGC
327   UNIT    DEBUGC
328   CALC    NC551→INFO(MO1)$CLS$54
329   SHOWA   NC551•1
-------------------------------------------------------------------------------- READINI
                                                                                TOOMUCH
331   UNIT    TOOMUCH
332   MODE    REWRITE
333   AT      2810
334   WRITE   ↑NO MEMORY (¬OT•MALLOT-MUSE•6¬1 WORDS).
335           -NEXT- TO RETRY THE TRANSFER
336           -DATA- TO STAY
337   DATA    READIN
338   *
                                                                                READIN
339   UNIT    READIN          $$ SET MSI TO FORMULA LENGTH FROM START
340   *
341   *       READ A FORMULA FROM THE KEYBOARD• ENCODE IT
342   *       AND FILL IT IN INFIX FROM N(START) TO N(START•MSI-1)
343   *
344   *       CODES↑! (LOW ORDER 6 BITS)
345   *       LETTERS --- VARIABLES
346   *       27..31 --- •-≠= POWER
347   *       32 IS FOLLOWED BY V→VALUE• CONSTANT
348   *       33 IS FOLLOWED BY (LOW ORDER 6 BITS)↑!
349   *       1 FOR (• 2... FOR FUNCTIONS IN ASCENDING TYPES
350   *       34 FOR )
351   *
352   ERASE
353   BASE
354   *
355   *       INITIALIZE CONSTANT TABLE
356   *
357   CALC    V(TABLE)→0      $$ BOTTOM CONSTANT
358           V(TABLE•1)→1    $$ NEXT CONSTANT
359           V(CPT→TABLE•2)→2    $$ TOP OF CONSTANT TABLE
360   INHIBIT ERASE
361   JOIN    DATELINE
362   AT      1010
363   WRITE   ↑ENTER A FORMULA• PLEASE.
364
365           A..Z FOR VARIABLES AND LETTER-CONSTANTS
366           DECIMAL CONSTANTS
367           • - ≠ = * / ** AS BINARY OPERATORS
```

```
          UNARY MINUS AND ( ) ALLOWED.
          LN,EXP,SIN,ASIN,COS,ACOS,TAN,ATAN
          COT,SEC,CSC,SINH,COSH,TANH,SQRT AS FUNCTIONS ALLOWED
AT        2001
WRITE     FORMULA=
AT        2410
WRITE     ↑W↑A↑R↑N↑I↑N↑G↑: MULTIPLICATION (* OR *)
                    TAKES PRECEDENCE OVER
                    DIVISION (= OR /)....
ARROW     2010
LONG      50
SPECS     NOOKNO
PUTD      ↑,↑,:,
BUMP
OK
JOIN      (JCOUNT=0)-(JCOUNT>END-START+1),NONEREAD,X,OVERAREA
JUDGE     CONTINUE
OPEN      N(M01)←END-JCOUNT,1) $$ POINTER TO BEGIN
OK
CALC      M02←START     $$ POINTER  TO CURRENT RECIPIENT
JUMP      READIN2       $$ FOR DISPATCH
*
*
*                                                                    READ2
UNIT      READ2
INHIBIT ERASE           $$ ANALYZE CONSTANT
CALC      N(M02)←↑,5↑, $$ FOR CONSTANT TO FOLLOW
          V(M02←M02+1)←0 $$ WILL TREAT SOLO . AS 0
          M03←0         $$ NUMBER OF DECIMAL PLACES
2DIG      BRANCH (N(M01)=↑.,↑:)-(N(M01)¬>↑:0↑,$AND$N(M01)¬<↑:9↑,),X,3D,1D
          BRANCH M03>0,3ERR,X $$ TOO MANY .
          M03←1         $$ MARK . AS FOUND
          BRANCH (M01←M01+1)¬<END,2DIG,3ADD
1D        V(M02)←V(M02)≠10←N(M01)-↑:0↑,
          M03←M03-(M03>0) $$ CHALK ONE PLACE UP
          BRANCH (M01←M01+1)¬<END,2DIG,X
3D        V(M02)←V(M02)≠10**((M03-1)(M03>0))
3ADD      M02←M02+1
          BRANCH 0
3ERR      M01←-1
JUMP      READIN2
*
*
*                                                                    READ21
UNIT      READ21        $$ FROM RETAIN ↑: SHIFTED LETTER IS CONSTANT
INHIBIT ERASE
CALC      N(M02)←↑:5↑:      $$ CONSTANT TO FOLLOW
          V(M02+1)←V(TABLE+2←N(M01+1)) $$ INFO
          M01←M01+2
          M02←M02+2
JUMP      READIN2
```

```
421  UNIT    READ31
422  INHIBIT ERASE           $$ ANALYZE LETTERS
423  CALC    BRANCH N(M01)=↑,C↑,$AND$M01<END-4,1C,X
424          BRANCH N(M01)=↑,F↑,$AND$M01<END-4,1E,X
425          BRANCH N(M01)=↑,L↑,$AND$M01<END-3,1L,X
426          BRANCH N(M01)=↑,T↑,$AND$M01<END-4,1T,X $$ FUNCTIONS
427  2LETTER N(M02)←N(M01)
428          BRANCH 3ONE
429  1C      BRANCH N(M01+1)=↑,O↑,$AND$N(M01+2)=↑,S↑,$AND$N(M01+3)=↑,H↑,,2N,X
430          BRANCH N(M01+1)=↑,O↑,$AND$N(M01+2)=↑,S↑,,2F,X
431          BRANCH N(M01+1)=↑,O↑,$AND$N(M01+2)=↑,T↑,,2J,X
432          BRANCH N(M01+1)=↑,S↑,$AND$N(M01+2)=↑,C↑,,2L,2LETTER
433  1E      BRANCH N(M01+1)=↑,X↑,$AND$N(M01+2)=↑,P↑,,2C,2LETTER
434  1L      BRANCH N(M01+1)=↑,N↑,,2B,2LETTER
435  1T      BRANCH N(M01+1)¬=↑,A↑,$OR$N(M01+2)¬=↑,N↑,,2LETTER,X
436          BRANCH N(M01+3)=↑,H↑,,2O,X
437          N(M02+1)←↑,H↑, $$ FOR TAN
438          BRANCH 3FOU
439  3FIV    M01←M01+1
440  3FOU    M01←M01+4
441  3LPAR1  M02←M02+1
442          BRANCH N(M01-1)¬=↑,(↑,,3ERR,X
443          N(M02-1)←33 $$ FOR FUN
444          BRANCH 3ADD
445  2B      N(M02+1)←↑,B↑, $$ FOR LN
446          M01←M01+3
447          BRANCH 3LPAR1
448  2C      N(M02+1)←↑,C↑, $$ FOR EXP
449          BRANCH 3FOU
450  2F      N(M02+1)←↑,F↑, $$ FOR COS
451          BRANCH 3FOU
452  2J      N(M02+1)←↑,J↑, $$ FOR COT
453          BRANCH 3FOU
454  2L      N(M02+1)←↑,L↑, $$ FOR CSC
455          BRANCH 3FOU
456  2N      N(M02+1)←↑,N↑, $$ FOR COSH
457          BRANCH 3FIV
458  2O      N(M02+1)←↑,O↑, $$ FOR TANH
459          BRANCH 3FIV
460  3ERR    M01←-2
461  3ONE    M01←M01+1
462  3ADD    M02←M02+1
463  JUMP    READIN2
464  *
465  *
466  *
```



```
467  UNIT    READ31
468  INHIBIT ERASE           $$ FUNCTIONS OR LETTER A
469  CALC    BRANCH N(M01+1)=↑,C↑,$AND$N(M01+2)=↑,O↑,$AND$N(M01+3)=↑,S↑,,2G,X
470          BRANCH N(M01+1)=↑,S↑,$AND$N(M01+2)=↑,I↑,$AND$N(M01+3)=↑,N↑,,2E,X
471          BRANCH N(M01+1)=↑,T↑,$AND$N(M01+2)=↑,A↑,$AND$N(M01+3)=↑,N↑,,2I,X
472  2LETTER N(M02)←N(M01)
473          BRANCH 3ONE
```

```
                2E      N(MO2+1)↛↑,E↑, $$ FOR ASIN
                        BRANCH 3FIV
                2G      N(MO2+1)↛↑,G↑, $$ FOR ACOS
                        BRANCH 3FIV
                2I      N(MO2+1)↛↑,I↑, $$ FOR ATAN
        3FIV    MO1↛MO1+5
                MO2↛MO2+1
                        BRANCH N(MO1-1)¬=↑,(↑,,3ERR,X
                        N(MO2-1)↛33 $$ FOR FUN
                        BRANCH 3ADD
        3ERR    MO1↛-2
        3ONE    MO1↛MO1+1
        3ADD    MO2↛MO2+1
        JUMP    READIN2

-------------------------------------------------------------------------------------------------------------------------------- READIN3
                                                                                                                          READ32
        UNIT    READ32
        INHIBIT ERASE           $$ LETTER OR FUNCTION S
        CALC    BRANCH N(MO1+1)=↑,I↑,$AND$N(MO1+2)=↑,N↑,$AND$N(MO1+3)=↑,H↑,,2M,X
                BRANCH N(MO1+1)=↑,I↑,$AND$N(MO1+2)=↑,N↑,,2D,X
                BRANCH N(MO1+1)=↑,Q↑,$AND$N(MO1+2)=↑,R↑,$AND$N(MO1+3)=↑,T↑,,2P,X
                BRANCH N(MO1+1)=↑,E↑,$AND$N(MO1+2)=↑,C↑,,2K,X
        2LETTER N(MO2)↛N(MO1)
                BRANCH 3ONE
        2D      N(MO2+1)↛↑,D↑, $$ FOR SIN
                BRANCH 3FOU
        2K      N(MO2+1)↛↑,K↑, $$ FOR SEC
                BRANCH 3FOU
        2M      N(MO2+1)↛↑,M↑, $$ FOR SINH
                BRANCH 3FIV
        2P      N(MO2+1)↛↑,P↑, $$ FOR SQRT
        3FIV    MO1↛MO1+1
        3FOU    MO1↛MO1+4
                MO2↛MO2+1
                        BRANCH N(MO1-1)¬=↑,(↑,,3ERR,X
                        N(MO2-1)↛33 $$ FOR FUN
                        BRANCH 3ADD
        3ERR    MO1↛-2
        3ONE    MO1↛MO1+1
        3ADD    MO2↛MO2+1
        JUMP    READIN2
        *
        *
        *
                                                                                                                          READ4
        UNIT    READ4
        INHIBIT ERASE           $$ TREAT SIGNS
        CALC    BRANCH N(MO1)=↑,+↑,,2PLUS,X
                BRANCH N(MO1)=↑,-↑,,2MINUS,X
                BRANCH N(MO1)=↑,≠↑,,2MULT,X
                BRANCH N(MO1)=↑,=↑,$OR$N(MO1)=↑,/↑,,2DIV,X
                BRANCH (N(MO1)=↑,*↑,)$AND$(MO1<END),1POW,X
                BRANCH N(MO1)=↑,)↑,,2RPAR,X
                BRANCH N(MO1)=↑,(↑,,2LPAR,2ERR
```

```
527             MO1→MO1+2
528             BRANCH 3ADD
529     2PLUS   N(MO2)→27 $$ FOR +
530             BRANCH 3ONE
531     2MINUS  N(MO2)→28 $$ FOR -
532             BRANCH 3ONE
533     2MULT   N(MO2)→29 $$ FOR ≠
534             BRANCH 3ONE
535     2DIV    N(MO2)→30 $$ FOR ≡
536             BRANCH 3ONE
537     2RPAR   N(MO2)→34 $$ FOR )
538             BRANCH 3ONE
539     2LPAR   N(MO2→MO2+1)→↑,A↑, $$ FOR (
540             N(MO2-1)→33 $$ FOR FUN
541             BRANCH 3ONE
542     3ERR    MO1→-2
543     3ONE    MO1→MO1+1
544     3ADD    MO2→MO2+1
545     *
546     *
547     *
                                                                        READIN2
548     ENTRY   READIN2
549     INHIBIT ERASE
550     JUMP    MO1<0,SYNTAX,X $$ ILLEGAL CHAR, ETC.
551     JUMP    MO2>MO1,OVERAREA,X $$ INFIX OVERFLOWS
552     JUMP    MO1¬<END,X,POSTFIX $$ REPEAT
553     JUMP    N(MO1)=SHIFT,READ21,X $$ ON RETAIN
554     JUMP    (MO1<END-4)≠((N(MO1)=↑,A↑,)=(N(MO1)=↑,S↑,)),READ32,X,READ31
555     JUMP    N(MO1)=↑,,↑,$OR$(N(MO1)¬>↑,0↑,$AND$N(MO1)¬<↑,9↑,),READ2,X
556     JUMP    N(MO1)¬>↑,A↑,$AND$N(MO1)¬<↑,Z↑,,READ3,READ4
557     *
558     *
559     *
                                                                        OVERAREA
560     UNIT    OVERAREA
561     MODE    REWRITE
562     AT      1001
563     ERASE   9
564     WRITE   ↑S↑Y↑S↑T↑E↑M ↑F↑A↑I↑L↑U↑R↑E --- AVAILABLE AREA EXCEEDED
565     ERASE   13
566     PAUSE
567     JUMP    ERROR
568     *
569     *
570     *
                                                                        SYNTAX
571     UNIT    SYNTAX
572     MODE    REWRITE
573     AT      1001
574     ERASE   9
575     WRITE   ↑E↑R↑R↑O↑R --- ILLEGAL SYNTAX IN EXPRESSION
576     ERASE   18
```

```
577   PAUSE
578   JUMP    ERROR
579   *
580   *
581   *                                                                    NONEREAD
582   UNIT    NONEREAD
583   MODE    REWRITE
584   AT      1001
585   ERASE   9
586   WRITE   ↑E↑R↑R↑O↑R --- NO FORMULA READ
587   ERASE   40
588   PAUSE
589   JUMP    ERROR
      ------------------------------------------------------------------------------------------------------------------------ POSTFIX1
                                                                           POSTFIX
591   UNIT    POSTFIX
592   *
593   *       FROM A FORMULA FORMATTED AS BY READIN, FROM START
594   *       TO BEFORE M02 LOCATED, CONSTRUCT POSTFIX FROM START
595   *       ON UP LOCATED. ↑INTO WHER(NODE) INSERT LOCATION
596   *       OF SPOT OF THIS ENTRY IN THE DISPLAYED INFIX.
597   *
598   *       DERIV--= ¬↑D TOP + 1 OF POSTFIX
599   *              SO SET AS RESULT↑.
600   *
601   AT      3028
602   WRITE   ↑F↑O↑R↑M↑A↑T↑T↑I↑N↑G
603   CALC    STOP←STACK      $$ NEXT ENTRY IN STACK
604           LPT←START       $$ NEXT ENTRY IN POSTFIX
605           LPARCT←0        $$ ¬$ OPEN (
606           M01←-1          $$ STATE↑↓ EXPECT OPERAND
607           XPT←FORLOC      $$ LOCATION OF SHOW CODE(↑INFIX)
608           XUS←1           $$ USE OF INFIX..
609           XSI←FORMAX      $$ WORDS IN SHOW CODE
610           WHE←0           $$ LOCATION OF INFIX CODE
611           MPT←START       $$ CURRENTLY TREATED ITEM IN INFIX
612   AT      505
613   DO      POSTFIX0,M03←START,END,1 $$ CYCLE THE ANALYZER
614   *
615   *
616   *                                                                    POSTFIX0
617   UNIT    POSTFIX0
618   *       POSTFIX DRIVER UNIT
619   JOIN    M01←POSTFIX1←POSTFIX2 $$ JOIN OPND OR OPRT ROUTINE
620   JOIN    MPT¬>M02,POSTFIX3,X     $$ QUIT WITH CLEANUP
621   JOIN    LPT>MPT←OVERAREA.X $$ BLEW IT
622   *
623   *
624   *                                                                    POSTFIX3
625   UNIT    POSTFIX3        $$ CLEAN UP
```

```
627   JOIN    LPARCT>0$OR$MO1=-1,SYNTAX,X $$ BLEW IT
628   CALC    DERIV=LPT          $$ BOTTOM OF DERIVATIVE
630           FORSIZE=XUS-1      $$ SET SIZE IN CHAR
631   JUMP    OPTIONS            $$ AND GO TO NEXT PAGE
632   *
633   *
634   *
                                                                          POSTFIX1
635   UNIT    POSTFIX1           $$ EXPECT AND PROCESS OPERAND
636   JOIN    N(MPT)=32,X,PF11,PF12,SYNTAX
637   JOIN    N(MPT)=27,PF13,PF14,PF15,SYNTAX $$ DISPATCH
638   *
639   *
640   *
                                                                          PF11
641   UNIT    PF11               $$ LEVEL 2 POSTFIX1
642   *       EXPECT OPERAND, FOUND CONSTANT
643   CALC    VAL=V(MPT+1)
644   JOIN    LOOKC              $$ ENTER INTO POSTFIX
645   CALC    MPT=MPT+2          $$ ADVANCE POINTERS, ETC.
646           LPT=(BPT=LPT)+1
647   JOIN    VAL=1$OR$VAL=2$OR$VAL=0,PF112,PF111
648   JOIN    INFIX4             $$ BUILD CODED INFIX
649   EXIT    2                  $$ FROM POSTFIX1
650   *
651   *
652   *
                                                                          PF12
653   UNIT    PF12               $$ LEVEL 2 POSTFIX1
654   *       EXPECT OPERAND, FOUND FUNCTION
655   WRITEC  (N(STOP)=N(MPT+1))-2,(,LN(,EXP(,SIN(,ASIN(,$$ STACK FUN
656           COS(,ACOS(,TAN(,ATAN(,COT(,SEC(,CSC(,SINH(,COSH(,TANH(,SQRT(,
657   CALC    MPT=MPT+2          $$ ADVANCE POINTERS,ETC
658           N(STOP+1)=WHE      $$ STACK LOCATION
659           STOP=STOP+2
660           LPARCT=LPARCT+1
661   JOIN    STOP->FORLOC,OVERSTAC,X
662   CALCS   N(STOP-2)-2,MO4=1,38,40,44,43,49,48,54,53,58,61,64,44,49,54,67
663   CALCS   N(STOP-2)-2,MO5=1, 2, 3, 3, 4, 3, 4, 3, 4, 3, 3, 3,4
664   CALC    WHE=WHE+MO5-(MO5-=1)
665   JOIN    (XUS=XUS+MO5)>10#XSI,OVERCORE,X $$ BLEW IT
666   MOVE    INLIST,MO4,N(XPT),XUS-MO5,MO5 $$ CODE INFIX
667   JOIN    MO4-=1,INFIX2,X $$ CODE ( FOR FUN
668   EXIT    2                  $$ FROM POSTFIX1

------------------------------------------------------------------------------ POSTFIX2
                                                                          PF111
670   UNIT    PF111              $$ SHOW NON 0 1 2 CONSTANT
671   CALCS   INT(LN(ABS(VAL))/LN(10))-1,V(MO1LOC)=2.3,3.2,4.1,6
672   SHOW    VAL,V(MO1LOC)
673   CALC    WHE=WHE+FORMLEN
674           MO1=0              $$ EXPECT OPERATOR
675   *
676   *
```

```
677  *
                                                                                PF112
678  UNIT    PF112           $$ SHOW 0 1 2
679  WRITEC  VAL-1*0*1*2*
680  CALC    WHE*WHE+1
681          MO1*0           $$ EXPECT OPERATOR
682  *
683  *
684  *
                                                                                PF13
685  UNIT    PF13            $$ LEVEL 2 POSTFIX 1
686  *       EXPECT OPERAND, FOUND LETTER
687  WRITE     <
688  SHOWA   N(MPT)*10
689  CALC    WHE*WHE+2       $$ ADVANCE POINTERS, ETC
690          N(BPT+LPT)*IINFO(N(MPT))$UNION$IWHER(WHE-1)$UNION$ITYPE(TYPEC)
691          N(BPT)*N(BPT)$UNION$IPRIN(TYPEC)$UNION$ILENG(1)
692          MPT*MPT+1
693          LPT*LPT+1
694          MO1*0
695  JUIN    INFIX3
696  EXIT    2               $$ FROM POSTFIX1
697  *
698  *
699  *
                                                                                PF14
700  UNIT    PF14            $$ LEVEL 2 POSTFIX1
701  *       EXPECT OPERAND, FOUND +
702  CALC    MPT*MPT+1       $$ IGNORE
703  EXIT    2               $$ FROM POSTFIX1
704  *
705  *
706  *
                                                                                PF15
707  UNIT    PF15            $$ LEVEL 2 POSTFIX1
708  *       EXPECT OPERAND, FOUND -
709  WRITE   -
710  CALC    N(STOP+1)*WHE   $$ STACK LOCATION
711          WHE*WHE+1
712          MPT*MPT+1
713          N(STOP)*-7  $$ STACK OPERATOR
714          STOP*STOP+2
715  JUIN    STOP->FORLOC,OVERSTAC,X $$ BLEW STACK
716  MOVE    INLIST,33,N(XPT),XUS,1 $$ CODE -
717  JUIN    (XUS*XUS+1)>10*XSI,OVERCORE,X $$ BLEW IT
718  EXIT    2               $$ FROM POSTFIX1
719  *
720  *
721  *
                                                                                OVERSTAC
722  UNIT    OVERSTAC
723  MODE    REWRITE
724  AT      1001
725  WRITE   ?????STATEMENT TOO COMPLEX --- EXPRESSION STACK OVERFLOWS
```

```
PAUSE   ,,
JUMP    ERROR
*
*
*
                                                                                                        POSTFIX2
UNIT    POSTFIX2        $$ EXPECT OPERATOR
CALC    OPR←26-N(MPT)   $$ READYA TO STACK -1 -2 .. FOR + - ...
JOIN    OPP←A,SYNTAX,PF21,SYNTAX,SYNTAX,PF22,PF22,PF22,PF22,PF22,SYNTAX
*
*
*
                                                                                                        PF21
UNIT    PF21
*       EXPECT OPERATOR, FOUND )
JOIN    LPARCT¬<0,SYNTAX,X $$ BLEW IT
CALC    OPR←1
        MPT←MPT+1
JOIN    PFSTACK $$ POP TO AND INCLUDING (
WRITE   )
CALC    WHE←WHE+1
JOIN    INFIX6          $$ CODE )
EXIT    2
*
*
*
                                                                                                        PF22
UNIT    PF22            $$ LEVEL 2  POSTFIX2
*       EXPECT AND FOUND OPERATOR
JOIN    PFSTACK $$ POP TO PRECEDENCE
CALC    N(STOP)←OPR
        N(STOP+1)←WHE
        MPT←MPT+1
        STOP←STOP+2
        MO1←-1
WRITEC  -2-OPR,+,-,*,/,↑,**,
CALC    WHE←WHE+1-(OPR=-5)
MOVE    INLIST,31=OPR,N(XPT),XUS,1-(OPR==5)
JOIN    (XUS←XUS+1-(OPR=-5))>10≠XSI,OVERCORE,X $$ BLEW IT
JOIN    STOP¬>FORLOC,OVERSTAC,X
EXIT    2               $$ FROM POSTFIX2

                                                                                                        OUTPUT
                                                                                                        PFSTACK
UNIT    PFSTACK
*
*       POP STACK UNTIL LOWER PRECEDENCE EMPTY OR ( POPPED
*
CALC    BRANCH STOP¬<STACK,Q,X $$ EMPTY...
4LOOP   BRANCH OPR>0,1POP,X $$ EXPECT )
        BRANCH N(STOP-2)>0,Q,X $$ BINARY OP CAN'T POP (
        BRANCH INT((1=OPR)/2)≥INT((1-N(STOP-2))/2),Q,X $$ PRECEDENCE
        BRANCH OPR=-3$AND$N(STOP-2)=-4,Q,X $$ MULT PRECEDES DIV
1POP    LOPR←-N(STOP-2)+TYPEADD-1+(N(STOP-2)=-7)
```

```
777             BRANCH N(STOP-2)<0,2NOFUN,X $$ OPERATOR POPS
778             LOPR←N(STOP-2),TYPELOG-2
779             BRANCH N(STOP-2)=1,3NOOUT,X $$ FUNCTION POPS
780   2NOFUN    N(LPT)←TYPE(LOPR)$UNION$IPRIN(LOPR)$UNION$ILENG(1)
781             N(LPT)←N(LPT)$UNION$1WHER(N(STOP-1))
782             LPT←LPT+1
783             BRANCH LPT>MPT,0,X $$ BLEW AREA
784   3NOOUT    STOP←STOP-2
785             LPARCT←LPARCT+(N(STOP)>0)
786             BRANCH (N(STOP)<0)$AND$(STOP>STACK),4LOOP,X
787   *
788   *
789   *
                                                                              OUTPUT
790   UNIT      OUTPUT
791   *
792   *         DISPLAY ENCODED INFIX
793   *
794   *         --- LOCATION SET BY EXTERNAL AT STATEMENT
795   *         XPT--- →D ENCODED INFIX
796   *         XSI--- LENGTH OF IT IN CHAR
797   *
798   CALC      M01←0           $$ OFFSET TO ACTUAL CODE
799   DO        OUTPUT1,M04←1,XSI,1 $$ CYCLE THE SHOW
800   *
801   *
802   *
                                                                              OUTPUT1
803   UNIT      OUTPUT1         $$ SHOW 10 CHAR OR UP TO NON 0 1 2 CON
804   DO        OUTPUT2,M05←M04,M04←9,1 $$ SCAN
805   JOIN      (M04-=M05),OUTPUT3,X $$ SHOW SPLIT OFF
806   JOIN      (M02=SHIFT),OUTPUT4,X $$ SHOW CONSTANT
807   CALC      M01←M01,M05-M04-1 $$ RESET OFFSET
808   EXIT      -(M04+M01¬>XSI) $$ AND QUIT IF NECESSARY
809   *
810   *
811   *
                                                                              OUTPUT2
812   UNIT      OUTPUT2         $$ LOCATE SHIFT (PRECEEDS CON)
813   CALC      M02←0
814   EXIT      -(M05+M01>XSI) $$ ONLY WITHIN STRING
815   MOVE      N(XPT),M05←M01,M02,10,1 $$ GET SINGLE CODE
816   EXIT      -(M02=SHIFT) $$ AND IF NOT KEEP GOING
817   *
818   *
819   *
                                                                              OUTPUT3
820   UNIT      OUTPUT3         $$ PRINT STRING ¬< 10 CHAR
821   CALC      M03←0
822   MOVE      N(XPT),M04←M01,M03,1,M05-M04
823   SHOWA     M03,M05-M04
824   *
825   *
826   *
```

```
827  UNIT    OUTPUT4          $$ SHOW CONSTANT
828  MOVE    N(XPT),M05+M01+1,M03+10,1
830  CALCS   INT(LN(ABS(V(TABLE+2+M03)))/LN(10))-1,V(M02LOC)+2,3,3,2,4,1,6
831  SHOW    V(TABLE+2+M03),V(M02LOC)
832  CALC    M01+M01+2
833  *
834  * ------------------------------------------------
835  *

 ---------------------------------------------------------------------------------------------------------------------------- INFIX
                                                                                                                          INFIX
837  UNIT    INFIX
838  *
839  *       TRANSLATE FRAGMENT INTO  ↑T↑U↑T↑O↑R INFIX
840  *
841  *       APT--- →↑D FRAGMENT TO BE INFIXED
842  *       XPT--- →↑D INFIX LOCATION IN CORE
843  *       XSI--- MAXIMAL INFIX SIZE IN WORDS
844  *              SET TO ACTUAL USAGE IN CHAR
845  *
846  CALC    XUS+1            $$ BEGIN AT FIRST INFIX POSITION
847  DO      INFIX1,BPT+APT-LENG(APT)+1,APT,1 $$ LOOP ON SHOW
848  CALC    XSI+XUS-1
849  *
850  *
851  *
                                                                                                                          INFIX1
852  UNIT    INFIX1
853  *       SHOW A SINGLE ITEM
854  DO      INFIX2,M01+1,LPAR(BPI),1 $$ SHOW (
855  JOIN    TYPE(BPT)-TYPEX-2,INFIX3,INFIX4,INFIX4,INFIX4,INFIX5
856  DO      INFIX6,M01+1,RPAR(BPI),1 $$ SHOW )
857  *
858  *
859  *
                                                                                                                          INFIX2
860  UNIT    INFIX2           $$ SHOW (
861  MOVE    INLIST,1,N(XPT),XUS,1
862  JOIN    (XUS+XUS+1)>10≠XSI,OVERCORE,X $$ BLEW IT
863  *
864  *
865  *
                                                                                                                          INFIX3
866  UNIT    INFIX3           $$ SHOW CONSTANT NAME
867  MOVE    INLIST,72,N(XPT),XUS+1 $$ BLANK
868  MOVE    INLIST,INFO(BPT)+4,N(XPT),XUS+1,1 $$ LETTER
869  JOIN    (XUS+XUS+2)>10≠XSI,OVERCORE,X $$ BLEW IT
870  *
871  *
872  *
                                                                                                                          INFIX4
873  UNIT    INFIX4           $$ MOVE 0 1 2 OR CAPITAL FOR CONSTANT
874  JOIN    INFO(BPT)>2,INFIX7,X $$ FOR SHIFT
875  MOVE    INLIST,2+INFO(BPT),N(XPT),XUS,1 $$ 0 1 2 LETTER
```

```
876   JOIN    (XUS*XUS+1)>10*XSI,OVERCORE,X $$ BLEW IT
877   *
878   *
879   *
                                                                                    INFIX5
880   UNIT    INFIX5          $$ NAME OF OPERATOR OR FUNCTION
881   CALCS   TYPE(BPT)=TYPEADD+1,MO2*32,33,34,35,36,33,38
882           40,44,43,49,48,54,53,58,61,64,44,49,54,67
883   CALCS   TYPE(BPT)=TYPEADD+4,MO3*1,2,1,2,3,3,4,3,4,3,4
884           3,3,3,4,4,4,4
885   *ASSERT MO2 IN INLIST TO OPERATOR, MO3 LENGTH
886   JOIN    (XUS*XUS+MO3)>10*XSI,OVERCORE,X $$ BLEW IT
887   MOVE    INLIST,MO2,N(XPT),XUS=MO3,MO3
888   *
889   *
890   *
                                                                                    INFIX6
891   UNIT    INFIX6          $$ SHOW )
892   MOVE    INLIST,71,N(XPT),XUS,1
893   JOIN    (XUS*XUS+1)>10*XSI,OVERCORE,X $$ BLEW IT
894   *
895   *
896   *
                                                                                    INFIX7
897   UNIT    INFIX7          $$ SHOW SHIFT
898   MOVE    INLIST,31,N(XPT),XUS,1
899   JOIN    (XUS*XUS+1)>10*XSI,OVERCORE,X $$ BLEW IT
900   *
901   *
902   *
                                                                                    OVERCORE
903   UNIT    OVERCORE
904   *
905   *       TERMINAL↑S BLEW INFIX REPRESENTATION
906   *
907   MODE    REWRITE
908   AT      1001
909   ERASE   9
910   WRITE   ↑S↑Y↑S↑T↑E↑M ↑F↑A↑I↑L↑U↑R↑E --- DISPLAY AREA OVERFLOWS
911   ERASE   14
912   PAUSE
913   JUMP    ERROR
914   *
915   * ----------------------------------------
916   *

-------------------------------------------------------------------------------- OPTIONS1
                                                                                    OPTIONS
918   UNIT    OPTIONS         $$ DISPLAY AND DISPATCH OPTIONS
919   ERASE
920   BASE
921   JOIN    DATELINE
922   AT      505
923   CALC    XSI*FORSIZE
```

```
923   JOIN    OUTPUT          $$ FLAG OPTB AND SHOW THEM FORMULA
927   AT      1010
928   WRITE   ↑AVAILABLE OPERATIONS ---
929   AT      1215
930   WRITE   [A] READ A NEW FORMULA
931
932           [B] EVALUATE THE FORMULA ABOVE
933
934           [C] DIFFERENTIATE THE FORMULA ABOVE
935
936           [D] ASSIGN A VALUE TO A LETTER CONSTANT
937   AT      2010
938   WRITE   ↑YOUR CHOICE ---
939   AT      2032
940   WRITE   -↑B↑A↑C↑K- TO CANCEL THE CHOICE
                                                                    ERROR1
941   ENTRY   ERROR1
942   JOIN    DATELINE
943   AT      2026
944   ERASE   5
945   ARROW   2026
946   LONG    1
947   SPECS   NOOKNO
948   MATCH   MO1,A,B,C,D,↑C  $$ SECRET OPTION
949   INHIBIT ERASE
950   JUMP    MO1,ERROR1,READIN,OPTB,OPTC,OPTD,OPT↑C
951   *
952   *
                                                                    OPTB
953   UNIT    OPTB            $$ EVALUATE THE FORMULA
954   CALC    PAR1←1
955   1LOOP   N(PAR1)←0777777777777777777 $$ ↑,UNUSED↑,
956           BRANCH (PAR1←PAR1+1)¬<26,1LOOP,X
957           PAR1←START
958   *
959   *
                                                                    OPTB1
960   ENTRY   OPTB1           $$ RUN DOWN FORMULA AND REQUEST VAL
961   INHIBIT ERASE
962   BASE
963   JUMP    TYPE(PAR1)¬=>TYPE0,OPTB2,X $$ NO LETTER
964   JUMP    N(INFO(PAR1))¬=0777777777777777777777,OPTB2,X $$ SET
965   JOIN    OPTB3
                                                                    OPTB2
966   ENTRY   OPTB2           $$ LOOPING OVER POSTFIX
967   INHIBIT ERASE
968   JUMP    (PAR1←PAR1+1)<DERIV,OPTB1,X
969   COMPUTE VAL,FORMULA,FORSIZE,PAR1      $$ GET VALUE
970   AT      525-PAR2≠185
971   WRITE   =
972   AT      WHERE+2
973   SHOWS   VAL
974   AT      3030
975   WRITE   -↑N↑E↑X↑T-
```

```
NEXT    PAR2,OPTIONS,RETAIN
*
*                                                                                    OPTB3
UNIT    OPTB3           $$ DRIVEN BY PAR1,PAR2 FOR OPT/RETAIN
BACK    PAR2,OPTIONS,RETAIN $$ RETURN POINT
AT      3010+PAR2*600
WRITE   ↑LET
ERASE   3
WRITE   =
ERASE   49
AT      3014+PAR2*600
WRITEC  INFO(PAR1)-2,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
ARROW   3018+PAR2*600
STORE   V(INFO(PAR1))
AT      3110+PAR2*500
WRITE   ↑E↑R↑R↑O↑R --- INVALID SYNTAX IN YOUR EXPRESSION
OK
*
*                                                                                    OPTD
UNIT    OPTD            $$ REPLACE VARIABLE BY VALUE
INHIBIT ERASE
BASE
BACK    OPTIONS
AT      2410
WRITE   ↑LET
ARROW   2414
LONG    1
SPECS   NOOKNO,NOCAPS
OK
JUDGE   (MO1*KEY-↑,A↑,,1)¬>1$AND$MO1¬<26,OK,IGNORE $$ GET NAME
AT      2418
WRITE   =
ARROW   2420            $$ VALUE
STORE   VAL
AT      2610
WRITE   ↑E↑R↑R↑O↑R --- ILLEGAL SYNTAX IN YOUR EXPRESSION
OK
CALC    MO2*0           $$ WILL HOLD REPLACEMENT INFO
        MO3*0           $$ WILL COUNT OCCURRENCES
        LPT*START       $$ THROUGH NODES
DO      OPTD1,MO6*START,DERIV,1 $$ CYCLE REPLACEMENT
JUMP    MO3=0,OPTIONS,X         $$ NONE
CALC    MO1*MO1+05500 $$ ADD A BLANK
        XUS*1
        MO2*INFO(MO2LOC)
DO      OPTD2,MO3*MO3+1,-1 $$ REPLACE IN INFIX
JUMP    OPTIONS

------------------------------------------------------------------------------------ OPTIONS2
                                                                                     OPTD1
UNIT    OPTD1           $$ REPLACE ONE OCCURRENCE IN POSTFIX
CALC    MO3*MO3+1
```

```
1027          BRANCH INFO(LPT)=MO1,Q,X $$ SEARCH
1028          MO3=MO3-1 $$ NO MORE
1029  EXIT    -(LPT->DERIV)
1030  JOIN    (MO2=0)(WHE=WHER(LPT)),LOOKC,X $$ FIX POSTFIX
1031  CALC    BRANCH MO2-=0,1DONT,X $$ ESTABLISH INFO
1032          MO2=N(LPT)$MASK$ZWHER $$ ALL BUT WHER
1033          MO4=FORMLEN-2  $$ USUAL OPENING SIZE
1034          BRANCH TYPE(LPT)=TYPEV,1SEAR,X
1035          MO4=0          $$ JUST OVERTYPE
1036  1SEAR   LPT=MO5=LPT+1  $$ DOWN THE POSTFIX
1037          BRANCH MO5->DERIV,Q,X
1038  2LOOP   XUS=WHER(MO5)  $$ LOCATION
1039          BRANCH XUS-<WHE,2NOWHE,X $$ DONT UPDATE
1040          N(MO5)=N(MO5)$MASK$ZWHER$UNION$IWHER(XUS+MO4)
1041  2NOWHE  BRANCH (MO5=MO5+1)<DERIV,2LOOP,Q
1042  1DONT   N(LPT)=MO2$UNION$IWHER(WHE)
1043          BRANCH 1SEAR
1044  EXIT    -(LPT->DERIV)
1045  *
1046  *
1047  *
```

OPTD2

```
1048  UNIT    OPTD2          $$ REPLACE IN INFIX ONE OCCURRENCE
1049  DO      OPTD3,XUS=XUS,FORSIZE,1
1050  JOIN    MO2>2,OPTD4,X  $$ SHIFT
1051  MOVE    INLIST,MO2+2,FORMULA,XUS=XUS+1,1
1052  *
1053  *
1054  *
```

OPTD3

```
1055  UNIT    OPTD3
1056  MOVE    FORMULA,XUS,MO4,9,2
1057  EXIT    -(MO4=MO1)
1058  *
1059  *
1060  *
```

OPTD4

```
1061  UNIT    OPTD4
1062  MOVE    INLIST,31,FORMULA,XUS,1
1063  *
1064  *
1065  *
```

OPTIONS3

OPT↑C

```
1066  UNIT    OPT↑C          $$ BYPASS ALL QUESTIONING
1067  CALC    MO1=REQUEST
1068  1LOOP   N(MO1)=N(MO1)$MASK$07070707070707070
1069          BRANCH (MO1=MO1+1)-<REQUEST+1,1LOOP,X
1070          N(REQUEST+2)=N(REQUEST+2)$MASK$07070707070707070000
1071  *
1072  *
1073  *
```

```
1076  ENTRY   OPTC          $$ DIFFERENTIATE AND QUESTION
1077  BASE
1078  BACK    OPTIONS
1079  AT      2410
1080  WRITE   ↑DIFFERENTIATE WITH RESPECT TO
1081  ARROW   WHERE↑2
1082  LONG    1
1083  SPECS   NOOKNO.NOCAPS
1084  OK
1085  JUDGE   (DVAR↑KEY-↑.A↑.↑1)¬>1$AND$DVAR¬<26.OK.IGNORE
1086  ERASE
1087  CALC    TDERIV↑DERIV-1
1088          TFORMEL↑START-1
1089          ITEM↑START
1090          MO1↑START
1091  1LOOP   BRANCH TYPE(MO1)¬>TYPE0.2NONE.X
1092          N(MO1)↑N(MO1)$MASK$ZTYPE$MASK$ZPRIN
1093          N(MO1)↑N(MO1)$UNION$IPRIN(TYPEX-(INFO(MO1)¬=DVAR))
1094          N(MO1)↑N(MO1)$UNION$ITYPE(PRIN(MO1))
1095  2NONE   BRANCH (MO1↑MO1↑1)<DERIV.1LOOP.X
1096  JUMP    (N(REQUEST↑2)$MASK$07)=07.X.DERIVE $$ NO QUESTIONS
1097  JOIN    DATELINE
1098  AT      505
1099  CALC    XSI↑FORSIZE
1100          XPT↑FORLOC
1101          N(REQUEST↑2)↑N(REQUEST↑2)$MASK$07777777777777777707
1102  JOIN    OUTPUT $$FORMULA
1103  AT      1005
1104  WRITE   ↑U =
1105  AT      1305
1106  WRITE   ↑D ↑U.
1107  AT      40.296
1108  SHOWA   DVAR.10
1109  AT      1605
1110  WRITE   ↑V =
1111  AT      1905
1112  WRITE   ↑D ↑V=
1113  AT      40.200
1114  SHOWA   DVAR.10
1115  AT      1031
1116  WRITE   ↑THIS ARROW ¬J WILL
1117          MARK THE ITEM IN THE
1118          FORMULA ABOVE. WHICH
1119          CAN BE DIFFERENTIATED.
1120  AT      1531
1121  WRITE   ↑IT WILL ↑7VISIT↑7 ALL
1122          SYMBOLS IN THE FORMULA.
1123          FOR EACH OPERATOR FIRST
1124          THE OPERANDS. THEN THE
1125          OPERATOR.
1126  AT      2224
1127  PLOT    126
1128  AT      3213
```

```
1133  WRITE   ↑ARE YOU READY ↑/ ---
1134  INHIBIT ERASE
1135  JUMP    OPTC1 $$ TO AVOID ERASING THE ARROW
1136  *
1137  *
1138  *
                                                                                OPTC1
1139  UNIT    OPTC1
1140  INHIBIT ERASE
1141  BASE
1142  BACK    OPTIONS
1143  JOIN    DATELINE
1144  HELP    ANSYES
1145  AT      2634
1146  ERASE   31
1147  ARROW   2633
1148  SPECS   NOOKNO,NOMARK
1149  ANSWER  YES
1150  JUMP    DERIVE
1151  NO
1152  *
1153  *
1154  *
                                                                              ANSYES
1155  ENTRY   ANSYES
1156  INHIBIT ERASE
1157  BASE
1158  NEXT    OPTC1
1159  AT      2633
1160  ERASE   1
1161  AT      2813
1162  WRITE   ↑TRY ↑7YES↑7 FOR AN ANSWER --- BE BRAVE.
1163          ↑PUSH =↑N↑E↑X↑T=.

                                                                                DERIVE1
                                                                              DERIVE
1165  UNIT    DERIVE
1166  *
1167  *       DIFFERENTIATE FORMULA ENTRY AT ITEM
1168  *
1169  ERASE
1170  BASE
1171  JOIN    DATELINE
1172  AT      505
1173  CALC    XSI←FORSIZE
1174          XPT←FORLOC
1175  JOIN    OUTPUT $$ FORMULA
1176  CALC    M01←WHER(ITEM)
1177          M01←M01-(M02←INT(M01/60))≠60
1178          OPR←TYPE(ITEM)
1179  AT      (6+M02+2(M02=0))≠100+M01+5
1180  WRITEC  M02=0,¬J,¬I,
1181  CALCS   OPR=2,DOPT←1,2,2,2,2,3,4,5,8,10,OPR←2
```

```
1186  CALC    TYP↑D↑U↑TYP↑D↑V↑1  $$ NOT 1 OR 0
1187          BRANCH DOPT<5$OR$DOPT>10,Q,X $$ SET COPT RIGHT
1188          BRANCH DOPT=8↑X↑1DIV↑1POW    $$ DISTRIBUTE
1189          BRANCH VALUE(TDERIV)↑Q↑X      $$ C↑U
1190          BRANCH VALUE(TDERIV-LENG(TDERIV))↑Q↑X
1191          BRANCH DOPT↑6↑Q↑Q             $$ ↑U↑V
1192  1DIV    BRANCH VALUE(TDERIV)↑Q↑X      $$ ↑U/C
1193          BRANCH VALUE(TDERIV-LENG(TDERIV))↑X↑2DIV
1194          BRANCH DOPT↑7↑Q↑Q             $$ C/↑U
1195  2DIV    BRANCH DOPT↑9↑Q↑Q             $$ ↑U/↑V
1196  1POW    BRANCH VALUE(TDERIV)↑Q↑X      $$ ↑U^C
1197          BRANCH VALUE(TDERIV-LENG(TDERIV))↑X↑2POW
1198          BRANCH DOPT↑11↑Q↑Q            $$ C^↑V
1199  2POW    DOPT↑12                       $$ ↑U^↑V
1200  *       DOPT NOW TO DIFFERENTIATION TYPE
1201  RETURN   $$ KEEPS THE NEXT CALC FROM BEING MERGED WITH ABOVE
1202  CALC    COPT↑NC(DOPT)
1203  JOIN    NC(COPT)↑DISPLAY1↑DISPLAY2↑DISPLAY4 $$ SET PAGE
1204  JOIN    NC(COPT)↑DERVAR↑DERUNA $$ BUILD DERIVATIVE
1205  CALC    APT↑TDERIV
1206          XPT↑WORKLOC
1207          XSI↑WORKMAX     $$ FOR DERIVATIVE
1208  JOIN    INFIX
1209  MOVE    N(REQUEST),DOPT↑M01↑10↑1
1210  CALC    WORKSIZ↑XSI
1211          ITEM↑ITEM↑1
1212          M02↑(M01$MASK$07)-1 $$ QUESTIONABLE↑/↑/
1213          M03↑(M01$MASK$077)-1
1214  CALCS   M02↑M01↑M01↑M03
1215  MOVE    M01↑10↑N(REQUEST),DOPT↑1 $$ RETURN COUNTER
1216  JUMP    M02=6↑INHIBIT↑X $$ DOESN↑T KNOW RULE
1217  JUMP    (M02=-1)-2*(ITEM<DERIV)↑RETAIN↑X↑DERIVE↑X
1218  JOIN    NC(COPT)↑QUEST1↑QUEST2↑QUEST4 $$ QUESTION AND REPEAT
1219  *
1220  *
1221  *
                                                                        INHIBIT
1222  UNIT    INHIBIT
1223  MODE    REWRITE
1224  AT      1010
1225  ERASE   9
1226  WRITE   ↑E↑R↑R↑O↑R --- INHIBITED DIFFERENTIATION RULE
1227  ERASE   15
1228  PAUSE
1229  JUMP    ERROR
1230  *
1231  *
1232  *
                                                                        DISPLAY1
1233  UNIT    DISPLAY1        $$ SHOW ↑U PROMPT ↑D↑U
1234  AT      1005
1235  WRITE   ↑U  =
1236  AT      1010
1237  *       SET FOR   CONSTANT FUN    BINARY
```

```
1237  JOIN    INFIX
1238  JOIN    OUTPUT $$ FOR DISPLAY
1239  AT      1305
1240  WRITE   ↑U ↑U=
1241  AT      40,296
1242  SHOWA   DVAR,10
1243  AT      3210
1244  WRITE   -↑H↑E↑L↑P- AVAILABLE
1245  *
1246  *
1247  *                                                                    DISPLAY2
1248  UNIT    DISPLAY2        $$ SHOW ↑U ↑D↑U SET  TYP↑D↑U
1249  JOIN    DISPLAY1        $$ SHOW ↑U PROMPT ↑D↑U
1250  AT      1310
1251  CALCS   NC(COP1)-1,APT↗TDERIV,TDERIV-LENG(TDERIV)
1252  CALC    TYP↑D↑U↗PRIN(APT) $$ SET ↑D↑U TYPE UP
1253          XSI↗WORKMAX
1254  JOIN    INFIX
1255  JOIN    OUTPUT
1256  *
1257  *
1258  *
                                                                    DERIVE2
                                                                    DISPLAY4
1260  UNIT    DISPLAY4        $$ SHOW ↑U ↑D↑U ↑V ↑D↑V SET TYP↑D↑U TYP↑D↑V
1261  JOIN    DISPLAY2        $$ SHOW ↑U ↑D↑U
1262  AT      1605
1263  WRITE   ↑V  =
1264  AT      1610
1265  CALC    APT↗TFORMEL
1266          XSI↗WORKMAX
1267  JOIN    INFIX
1268  JOIN    OUTPUT
1269  AT      1905
1270  WRITE   ↑D ↑V=
1271  AT      40,200
1272  SHOWA   DVAR,10
1273  AT      1910
1274  CALC    APT↗TDERIV
1275          TYP↑D↑V↗PRIN(APT) $$ SET TYPE ↑D↑V
1276          XSI↗WORKMAX
1277  JOIN    INFIX
1278  JOIN    OUTPUT
1279  *
1280  *                                                                    DERVAR
1281  UNIT    DERVAR
1282  *       DIFFERENTIATE A VARIABLE OR CONSTANT
1283  CALC    N(TFORMEL↗TFORMEL+1)↗N(ITEM) $$ COPY FORMEL
1284          LPT↗TDERIV↗TDERIV+1 $$ SET FOR COPY A ↑71↑7 OR ↑70↑7
1285          VAL↗=(OPR<2) $$ THE DERIVATIVE, BY TYPE
```

```
1286            WHE*WHER(TFORMEL)
1287    INHIBIT ERASE
1288    JUMP    TDERIV>END,OVERFORM,X $$ BLEW AREA
1289    JOIN    LOOKC $$ TO ENTER THE DERIVATIVE
1290    CALC    APT*TDERIV
1291            XPT*WORKLOC
1292            XSI*WORKMAX     $$ SET TO INFIX DERIVATIVE
1293    JOIN    INFIX
1294    CALC    WORKSIZ*XSI     $$ SET ITS LENGTH
1295    *
1296    *
1297    *                                                                       DERUNA
1298    UNIT    DERUNA
1299    *       DIFFERENTIATE AN OPERATOR
1300    CALC    PAR1*COPT+NC(COPT+2)+4 $$ TO BEGIN COMMAND LIST
1301            TYP+D+U*(PRIN(TDERIV-LENG(TDERIV))=TYPE0)+1
1302            TYP+D+V*(PRIN(TDERIV)=TYPE0)+1
1303    DO      DERUNA1,PAR2*NC(PAR1-1),1,-1 $$ CYCLE THEM
1304    CALC    OPR*TYPE(ITEM) $$ RESET
1305            APT*TFORMEL     $$ AND STRUCTURE FORMULA
1306    JOIN    OPR-TYPEADD-1,ADD,SUB,MUL,DIV,POW,UNARY
1307    *
1308    *
1309    *                                                                       DERUNA1
1310    UNIT    DERUNA1         $$ ONE COMMAND
1311    CALC    INS*NC(PAR1+1)  $$ -1 APPLIES TO TOP, 0 TO BELOW
1312            APT*TDERIV-(1+INS)*LENG(TDERIV) $$ POINT TO IT
1313            MO2*NC(PAR1)    $$ SAVE
1314    CALCC   NC(PAR1)=4,X,OPR*TYPEMIN,OPR*NC(PAR1+2) $$ FUNCTION
1315            FROM*NC(PAR1+2) $$ INSERTION FROM FORMEL
1316            VAL*NC(PAR1+2)   $$ INSERTION OF A NODE
1317    CALCS   NC(PAR1)-5,PAR1*PAR1+2,PAR1+3 $$ ADVANCE
1318    JOIN    MO2*ADD,SUB,MUL,DIV,POW,UNARY,UNARY,INSERT,NODE
1319    *       THAT BUILDS THE DERIVATIVE
------------------------------------------------------------------------------ QUESTION1
                                                                        QUEST1
1321    UNIT    QUEST1          $$ INTERROGATE ON VARIABLE/CONSTANT
1322    INHIBIT ERASE
1323    BASE
1324    HELP    HELP1
1325    ARROW   1310
1326    SPECS   NODEFNS
1327    PUTD    *,*,:,
1328    STORE   VAL
1329    AT      1610
1330    WRITE   †E†R†R†O†R --- ILLEGAL SYNTAX IN YOUR EXPRESSION
1331    ANSV    2-DOPT
1332    JUMP    NEXTITEM
1333    WRONGV  DOPT-1
1334    AT      1610
1335    WRITEC  1-DOPT
1336            *YOU ARE DIFFERENTIATING A CONSTANT
```

```
1339        AT      1610
1340        WRITE   ↑NEED -↑H↑E↑L↑P-↑/↑/↑/
1341        *
1342        *
                                                                                HELP1
1343        UNIT    HELP1
1344        BASE
1345        CALC    HELPS↑HELPS+1
1346        AT      1310
1347        MODE    REWRITE
1348        SHOW    2-DOPT,2
1349        ERASE   52
1350        AT      1910
1351        WRITEC  1-DOPT
1352                ↑THE DERIVATIVE OF A CONSTANT IS 0.
1353                ↑THE DERIVATIVE OF THE VARIABLE IS 1.
1354        MOVE    N(REQUEST),DOPT,M01,10,1
1355        CALC    M01↑(M01$MASK$077)+1 $$ RESET -$ QUESTIONS
1356        MOVE    M01,10,N(REQUEST),DOPT,1
1357        *
1358        *
                                                                                NEXTITEM
1359        ENTRY   NEXTITEM        $$ CONTINUE
1360        INHIBIT ERASE
1361        BASE
1362        AT      3210
1363        ERASE   20
1364        WRITE   -↑N↑E↑X↑T-
1365        NEXT    ITEM<DERIV,DERIVE,RETAIN
1366        *
1367        *
                                                                                QUEST2
1368        UNIT    QUEST2          $$ INTERROGATE UNARY OPERATOR
1369        INHIBIT ERASE
1370        BASE
1371        HELP    HELP2
1372        AT      1510
1373        WRITE   ↑U IS A FUNCTION OF THE DIFFERENTIATION VARIABLE  ≤
1374        SHOWA   DVAR,10
1375        WRITE   ,
1376                ↑D↑U IS ITS DERIVATIVE.
1377        AT      1805
1378        WRITE   ↑WHAT IS THE GENERAL DIFFERENTIATION RULE↑/
1379
1380                ↑D [
1381        AT      40,184
1382        SHOWA   DVAR,10
1383        AT      2008
1384        WRITEC  DOPT-14,-↑U,LN(↑U),EXP(↑U),SIN(↑U),ASIN(↑U),COS(↑U),
1385                ACOS(↑U),TAN(↑U),ATAN(↑U),COT(↑U),SEC(↑U),CSC(↑U),
1386                SINH(↑U),COSH(↑U),TANH(↑U),SQRT(↑U)
1387        AT      2015
1388        WRITE   ] =
```

```
1387  AT      2119
1390  WRITE   --- USE ONLY ↑U, ↑D↑U IN THIS ANSWER↑.
1391  DO      RANDOM,PAR2↗1,5,1 $$ RANDOMIZE
1392  ARROW   2019
1393  LONG    40
1394  SPECS   NOCAPS
1395  PUT     DU=↑DU
1396  PUT     U=↑U
1397  PUTD    ,↗,↓,
1398  BUMP    BFGJKMVW
1399  BUMP    Y7
1400  STOREA  ↑DFORM,↑DSIZE↗JCOUNT $$ JUDGE COPY
1401  STORE   ↑DVAL           $$ CORRECT↑/
1402  AT      2210
1403  WRITE   ↑E↑R↑R↑O↑R --- ILLEGAL SYNTAX OR QUANTITY IN EXPRESSION
1404  OK
1405  CALC    ↑DPT,↑SPT↗0      $$ FORCE A COMPILE
1406  COMPUTE VAL,NC(COPT+3),NC(COPT+1),↑SPT $$ SHOULD
1407  JUMP    ↑DVAL=VAL,QUESTA,X $$ OK
1408  DO      JUDGE↑S,PAR1↗1,2
1409  AT      2210
1410  WRITEC  PAR1-2
1411          ↑YOU MIGHT HAVE FORGOTTEN ↑D↑U.
1412          ↑YOU MIGHT HAVE FALSELY ASSUMED THAT ↑D↑U=0.
1413          ↑NEED -↑H↑E↑L↑P-↑/↑/↑/
1414  JUDGE   NO

--------------------------------------------------------------------------------------------------------------- QUESTION2
                                                                                                          HELP2
1416  UNIT    HELP2
1417  AT      2019
1418  ERASE   46
1419  AT      2019
1420  SHOWA   NC(COPT+3),NC(COPT+1)
1421  GOTO    HELP41
1422  *
1423  *
1424  *

                                                                                                          QUEST4
1425  UNIT    QUEST4          $$ INTERROGATE BINARY OPERATOR
1426  INHIBIT ERASE
1427  BASE
1428  HELP    HELP4
1429  AT      2110
1430  WRITE   ↑U, ↑V ARE FUNCTIONS OF THE DIFFERENTIATION VARIABLE  ≤
1431  SHOWA   DVAR,10
1432  WRITE   .
1433          ↑D↑U, ↑D↑V ARE THEIR DERIVATIVES, AND
1434          C IS CONSIDERED AN ARBITRARY CONSTANT.
1435  AT      2505
1436  WRITE   ↑WHAT IS THE GENERAL DIFFERENTIATION RULE↑/
1437
1438          ↑D [
1439  AT      40,72
1440  SHOWA   DVAR,1↗
```

```
1442  WRITEC  DOPT-4,↑U↑V,↑U-↑V,C↑V,↑U↑V,C↑↑,↑U↑V↑↑
1443  AT      2711
1444  WRITE   ] =
1445  AT      2815
1446  WRITE   --- USE ONLY C, ↑U, ↑V, ↑D↑U, ↑D↑V IN THIS ANSWER↑.
1447  DO      RANDOM,PAR2↑1,5,1
1448  ARROW   2715
1449  LONG    40
1450  SPECS   NOCAPS
1451  BUMP    !$A
1452  PUT     LN=A
1453  PUT     DU=:
1454  PUT     DV=$
1455  PUT     U=↑U
1456  PUT     V=↑V
1457  PUTD    ,↑,↑,,
1458  BUMP    BDEFGHIJ
1459  BUMP    LMNOPQR
1460  BUMP    STWXYZ
1461  PUT     !=↑D↑U
1462  PUT     $=↑D↑V
1463  PUT     A=LN
1464  STOREA  ↑DFORM,↑DSIZE↑JCOUNT $$ JUDGE COPY
1465  STORE   ↑DVAL $$ CORRECT↑/↑/
1466  AT      2910
1467  WRITE   ↑E↑R↑R↑O↑R --- ILLEGAL SYNTAX OR QUANTITY IN EXPRESSION
1468  OK
1469  CALC    ↑DPT↑↑SPT↑0 $$ FORCE COMPILATION
1470  COMPUTE VAL,NC(COPT+3),NC(COPT+1),↑SPT $$ SHOULD
1471  *       COULD CHECK SIMPLE HERE ....
1472  JUMP    ↑DVAL=VAL,QUESTA,X $$ MATCH
1473  CALCS   DOPT-5,PAR1↑1,3,1,3,1,1,1,3,1 $$ SET TO CYCLE OVER JUDGE↑S
1474  CALCS   DOPT-5,STOP↑5,5,5,5,5,3,5,3,5,5
1475  DO      JUDGE↑S,PAR1↑PAR1,STOP,1 $$ TRY OUT SOME WRONGS
1476  AT      2910
1477  WRITEC  PAR1+2(PAR1-<STOP)
1478          ↑YOU MIGHT HAVE FORGOTTEN ↑D↑U.
1479          ↑YOU MIGHT HAVE FALSELY ASSUMED THAT ↑D↑U = 0.
1480          ↑YOU MIGHT HAVE FORGOTTEN C.
1481          ↑YOU MIGHT HAVE FORGOTTEN ↑D↑V.
1482          ↑YOU MIGHT HAVE FALSELY ASSUMED THAT ↑D↑V = 0.
1483          ↑NEED -↑H↑E↑L↑P-↑/↑/↑/
1484  JUDGE   NO
1485  *
1486  *
1487  *                                                                    HELP4

1488  UNIT    HELP4
1489  AT      2715
1490  ERASE   50
1491  AT      2715
1492  SHOWA   NC(COPT+3),NC(COPT+1)
1493  AT      2810
1494  ERASE   55
```

```
1495  AT      2810
1496  WRITEC  DOPT-5
1497          ↑THE DERIVATIVE OF A SUM IS THE SUM OF THE DERIVATIVES.
1498          ↑A CONSTANT CAN BE FACTORED OUT.
1499          ↑PRODUCT RULE.
1500          ↑FACTOR OUT THE CONSTANT AND USE THAT 1/↑V = ↑V↑-↑1.
1501          ↑A CONSTANT CAN BE FACTORED OUT.
1502          ↑QUOTIENT RULE.
1503          ↑CONSTANT EXPONENT.
1504          ↑CONSTANT BASE, VARIABLE EXPONENT.
1505          ↑SORRY ABOUT THAT...
1506  *
1507  *
1508  *
                                                                        HELP41
1509  ENTRY   HELP41
1510  INHIBIT ERASE
1511  BASE
1512  MOVE    N(REQUEST),DOPT,M01,10,1
1513  CALC    HELPS↑=HELPS↑+1
1514          M01↑=(M01$MASK$077)↑+1 $$ RESET #$ QUESTIONS
1515  MOVE    M01,10,N(REQUEST),DOPT,1
1516  AT      3210
1517  ERASE   20
1518  WRITE   -↑N↑E↑X↑T-
1519  NEXT    QUESTA
1520  *
1521  *
                                                                        RANDOM
1522  UNIT    RANDOM          $$ SET V(PAR2) TO RANDU
1523  RANDU   V(PAR2)
                                                                        QUESTION3
                                                                        JUDGE↑S
1525  UNIT    JUDGE↑S         $$ JUDGE WRONG STRUCTURE FORMULA
1526  *
1527  *       EXIT FROM DO LOOP ON MATCH
1528  *
1529  *       PAR1--- 1 ↑D↑U↑=1 ↑↑FORGOT ↑D↑U↑↑
1530  *       PAR1--- 2 ↑D↑U↑=0 ↑↑MAY NOT BE 0↑↑
1531  *       PAR1--- 3 C ↑=1 ↑↑FORGOT C↑↑
1532  *       PAR1--- 4 ↑D↑V↑=1 ↑↑FORGOT ↑D↑V↑↑
1533  *       PAR1--- 5 ↑D↑V↑=0 ↑↑MAY NOT BE 0↑↑
1534  *
1535  CALCC   PAR1=2,↑D↑U↑=1,↑D↑U↑=0,C↑=1,↑D↑V↑=1,↑D↑V↑=0
1536  COMPUTE ↑DVAL,↑DFORM,↑DSIZE,↑DPT $$ GIVEN ANSWER
1537  COMPUTE VAL↑=NC(COPT↑+3),NC(COPT↑+1),↑SPT $$ CHECK
1538  DO      RANDOM↑,PAR2↑=1↑,5↑,1 $$ NEW SET
1539  EXIT    -(↑DVAL=VAL) $$ QUIT NEW ATTEMPT
1540  *
1541  *
1542  *
                                                                        QUESTA
1543  UNIT    QUESTA          $$ INTERROGATE ALGEBRA
1544  AT      2110-(DOPT<13)↑=700
```

```
1547  ERASE   35
1548  AT      2210-(DOPT<13)*700
1549  WRITE     =
1550  AT      3210
1551  WRITE   =↑H↑E↑L↑P= AVAILABLE
1552  AT      3230
1553  ERASE   6
1554  *
1555  *
1556  *
                                                                    QUESTA1
1557  ENTRY   QUESTA1
1558  INHIBIT ERASE
1559  BASE
1560  MODE    WRITE
1561  HELP    HELPA
1562  AT      2214-(DOPT<13)*700
1563  ERASE   51
1564  DO      RANDOM,PAR2=1,26,1 $$ RANDOMIZE
1565  ARROW   2215-(DOPT<13)*700
1566  LONG    72
1567  SPECS   NOCAPS
1568  PUTD    ,=,,,,=
1569  STORE   ↑DVAL
1570  AT      2415-(DOPT<13)*700
1571  WRITE   ↑E↑R↑R↑O↑R --- ILLEGAL SYNTAX IN YOUR EXPRESSION
1572  PAUSE
1573  JUMP    KEY=HELP,HELPA,X
1574  AT      2415-(DOPT<13)*700
1575  ERASE   43
1576  JUMP    QUESTA1
1577  OR
1578  CALC    PAR2=0          $$ FORCE COMPILE
1579  COMPUTE VAL,WORKFOR,WORKSIZ,PAR2 $$ SHOULD
1580  JUMP    =DVAL=VAL,NEXTITEM,X
1581  AT      2415-(DOPT<13)*700
1582  WRITE   =NEED =↑H↑E↑L↑P=↑/↑/↑/
1583  PAUSE
1584  JUMP    KEY=HELP,HELPA,X
1585  AT      2415-(DOPT<13)*700
1586  ERASE   14
1587  JUMP    QUESTA1
1588  *
1589  *
1590  *
                                                                    HELPA
1591  UNIT    HELPA
1592  INHIBIT ERASE
1593  AT      2215-(DOPT<13)*700
1594  ERASE   150
1595  AT      2215-(DOPT<13)*700
1596  CALC    XSI=WORKSIZ
1597          XPT=WORKLOC
```

```
1598            HELPA=HELPA+1
1599    JOIN    OUTPUT
1600    JUMP    NEXTITEM

------------------------------------------------------------------------------------------------------------------------ RETAIN
                                                                                                                  RETAIN
1602    UNIT    RETAIN
1603    *
1604    *       DISPLAY AND DISPATCH OPTIONS POST DIFF
1605    *
1606    ERASE
1607    BASE
1608    JOIN    DATELINE
1609    AT      510
1610    WRITE   ↑FORMULA↑:
1611    AT      605
1612    CALC    XSI=FORMAX
1613            XPT=FORLOC
1614            APT=TFORMEL     $$ NEW INFIX, FOR SIMPLIFICATION
1615    JOIN    INFIX
1616    CALC    FORSIZE=XSI
1617    JOIN    OUTPUT          $$ AND SHOW THE FORMULA
1618    AT      910
1619    WRITE   ↑DERIVATIVE WITH RESPECT TO  ↑;
1620    AT      WHERE-2
1621    SHOWA   DVAR,10
1622    AT      1005
1623    CALC    XSI=WORKSIZ
1624            XPT=WORKLOC
1625            PAR2=0
1626    JOIN    OUTPUT
1627    AT      1510
1628    WRITE   ↑AVAILABLE OPERATIONS ---
1629    AT      1715
1630    WRITE   [A] READ A NEW FORMULA
1631
1632            [B] EVALUATE THE FORMULA ABOVE
1633
1634            [C] EVALUATE THE DERIVATIVE ABOVE
1635
1636            [D] RETAIN THE FORMULA ABOVE
1637
1638            [E] RETAIN THE DERIVATIVE ABOVE
1639    AT      2710
1640    WRITE   ↑YOUR CHOICE ---
1641    AT      2732
1642    WRITE   -↑B↑A↑C↑K- TO CANCEL THE CHOICE
1643    CALC    MO1=REQUEST
1644            BRANCH (N(MO1+2)$MASK$070=070,Q,X $$ NOT YETASKED
1645    1LOOP   N(MO1)=N(MO1)$MASK$0707070707070707070
1646            N(MO1)=N(MO1)$UNION$(N(MO1)$CLS$57)
1647            BRANCH (MO1=MO1+1)-<REQUEST+2,1LOOP,X

                                                                                                                  RETAIN1
1648    ENTRY   RETAIN1
1649    JOIN    DATELINE
```

```
1651  ERASE   5
1652  ARROW   2726
1653  LONG    1
1654  SPECS   NOOKNO
1655  MATCH   MO1,A,B,C,D,E
1656  INHIBIT ERASE
1657  JUMP    MO1,RETAIN1,ERROR,OPTB,RETAINC,RETAIND,RETAIND
1658  *
1659  *
1660  *
                                                                        RETAINC
1661  UNIT    RETAINC         $$ EVALUATE THE DERIVATIVE
1662  CALC    PAR1+1
1663  1LOOP   N(PAR1)+07777777777777777777 $$ ↑,UNUSED↑,
1664          BRANCH (PAR1+PAR1+1)¬<26,1LOOP,X
1665          PAR1+DERIV
1666  *
1667  *
1668  *
                                                                        RETAINC1
1669  ENTRY   RETAINC1        $$ RUN DOWN FORMULA AND REQUEST VAL
1670  INHIBIT ERASE
1671  JUMP    TYPE(PAR1)¬>TYPE0,RETAINC2,X $$ NO LETTER
1672  JUMP    N(INFO(PAR1))¬=0777777777777777777777,RETAINC2,X $$ SET
1673  JOIN    OPTB3           $$ DUAL ACCESS UNIT
                                                                        RETAINC2
1674  ENTRY   RETAINC2           $$ LOOPING OVER POSTFIX
1675  INHIBIT ERASE
1676  JUMP    (PAR1+PAR1+1)¬<TDERIV,RETAINC1,X
1677  COMPUTE VAL,WORKFOR,WORKSIZ,PAR1        $$ GET VALUE
1678  AT      945
1679  WRITE   =
1680  AT      WHERE+2
1681  SHOWS   VAL
1682  AT      3030
1683  WRITE   =↑A↑E↑X↑I-
1684  NEXT    RETAIN
1685  *
1686  *
1687  *
                                                                        RETAIND
1688  UNIT    RETAIND         $$ KEEP THE FORMULA
1689  INHIBIT ARROW
1690  ARROW   101             $$ PHONY
1691  PRESS   NEXT
1692  LOADA   N(MO1+FORLOC+(MO1-3)(WORKLOC-FORLOC)),N(MO1+FORMAX)
1693  BUMP
1694  OPEN    N(MO1+END-JCOUNT+1)
1695  OK
1696  CALC    MO2+START       $$ NOW READYING FOR READIN2
1697          CPT+TABLE+2
1698  JUMP    (JCOUNT>END-START+1),OVERAREA,X $$ BLEW IT
1699  JUMP    READIN2
```

```
                                                
```

```
*---  MUST BUILD DIFFERENCE LOCALLY
JOIN  BUILD $$ TO BUILD THE DIFFERENCE
*
*
*
```

SUB1

```
UNIT  SUB1          $$ LEVEL 2 (SUB)
*     FORM -SUBTRAHEND IN SUB SEQUENCE
CALC  MPT=RPT
      OPR=TYPEMIN
JOIN  UMOVEDO $$ TO BUMP MINUEND
CALC  APT=MPT
JOIN  UNARY $$ TO FORM - SUBTRAHEND
EXIT  2             $$ FROM SUB
*
* ----------
*
```

MUL

```
UNIT  MUL           $$ RESETS TOPS
*
*     BUILD PRODUCT OF TWO FRAGMENTS
*
*     APT--- =↑D MULTIPLIER (MULTIPLICAND IS BELOW IN STACK)
*            IS SET TO PRODUCT
*
*---  SET SECOND POINTER
CALC  BPT=APT-LENG(APT) $$ BPT =↑D MULTIPLICAND
*---  MULTIPLICAND=0 OR MULTIPLIER = 1↑/
JOIN  PRIN(BPT)=TYPE0$OR$PRIN(APT)=TYPE1,ADD1,X $$ BUMP MULTIPLIER
*---  MULTIPLICAND=1 OR MULTIPLIER = 0↑/
JOIN  PRIN(APT)=TYPE0$OR$PRIN(BPT)=TYPE1,ADD2,X $$ BUMP MULTIPLICAND
*---  BUILD PRODUCT LOCALLY
CALC  OPR=TYPEMUL
JOIN  BUILD $$ TO BUILD THE PRODUCT
```

DIV PO

DIV

```
UNIT  DIV           $$ RESETS TOPS
*
*     BUILD QUOTIENT OF TWO FRAGMENTS
*
*     APT--- =↑D DIVISOR (DIVIDEND IS BELOW IN STACK)
*            IS SET TO QUOTIENT
*
*---  SET SECOND POINTER
CALC  BPT=APT-LENG(APT) $$ BPT =↑D DIVIDEND
*---  DIVIDEND = 0 OR DIVISOR = 1↑/
JOIN  PRIN(BPT)=TYPE0$OR$PRIN(APT)=TYPE1,ADD1,X $$ BUMP DIVISOR
*---  BUILD QUOTIENT LOCALLY
CALC  OPR=TYPEDIV
JOIN  BUILD $$ TO BUILD QUOTIENT
*
* ----------
*
```

```
                                                                                                                    POW
1806  UNIT    POW             $$ RESETS TOPS
1803  *
1804  *       BUILD BASE FRAGMENT TO POWER OF FRAGMENT
1805  *
1806  *       APT--- ¬↑D EXPONENT (BASE IS BELOW IN STACK)
1807  *              IS SET TO RESULT
1808  *
1809  *---    SET SECOND POINTER
1810  CALC    BPT←APT-LENG(APT) $$ BPT ¬↑D BASE
1811  *---    0↑^ANY↑v, 1↑^ANY↑v, ANY^1 ↑! DROP EXPONENT↑/
1812  JOIN    PRIN(BPT)=TYPE0$OR$PRIN(BPT)=TYPE1$OR$PRIN(APT)=TYPE1,ADD1,X
1813  *---    ANY^0 (BUT 0^0↑.) ↑! SET 1↑/
1814  JOIN    PRIN(APT)=TYPE0,POW1,X
1815  *---    MUST BUILD POWER LOCALLY
1816  CALC    OPR←TYPEPOW
1817  JOIN    BUILD $$ TO BUILD POWER
1818  *
1819  *
1820  *
                                                                                                                    POW1
1821  UNIT    POW1            $$ LEVEL 2 (POW)
1822  *       SET RESULT TO 1
1823  CALC    MPT←BPT
1824  JOIN    GMOVEDO $$ TO BUMP FRAGMENT(BPT)
1825  CALC    APT←LPT←MPT
1826          VAL←1
1827  JOIN    LOOKC $$ TO BUILD RESULT OF 1 OVER ORIGINAL 0
1828  EXIT    2               $$ FROM POW
1829  *
1830  * -------------------------------------------------
1831  *

  ----------------------------------------------------------------------------------------------------------- UNARY
                                                                                                                    UNARY
1833  UNIT    UNARY           $$ RESETS TOPS
1834  *
1835  *       APPLY UNARY OPERATOR TO A FRAGMENT
1836  *
1837  *       APT--- ¬↑D ARGUMENT
1838  *              SET TO RESULT
1839  *       OPR--- TYPE OF OPERATOR
1840  *
1841  *---    APPLYING INVERSE FUNCTIONS↑/
1842  JOIN    OPR=TYPEMIN$AND$PRIN(APT)=TYPEMIN,UNARY1,X
1843  JOIN    OPR=TYPEMIN$OR$OPR>TYPEATN$OR$ABS(OPR-PRIN(APT))¬=1,UNARY2,X
1844  JOIN    INT(OPR/2)=OPR/2$AND$PRIN(APT)>OPR,UNARY1,X
1845  JOIN    INT(OPR/2)¬=OPR/2$AND$PRIN(APT)<OPR,UNARY1,X
1846  JOIN    UNARY2
1847  *---    IF NOT, INSERT.
1848  *
1849  *
1850  *
                                                                                                                    UNARY1
1851  UNIT    UNARY1          $$ LEVEL 2 (UNARY)                                                                                  9
```

```
1854  JOIN    OPR=TYPEMIN$AND$PRIN(APT-LENG(APT)+1)>TYPEV,SPAREN,X
1855  CALC    N(APT)←N(APT)$MASK$ZPRIN$UNION$IPRIN(PRIN(APT-LENG(APT)+1))
1856          N(APT)←N(APT)$MASK$ZLENG$UNION$ILENG(LENG(APT)-1)
1857          MPT←APT-LENG(APT)
1858          N(MPT)←ILENG(1)
1859  JOIN    DMOVEDO $$ TO BUMP THE PREFIX
1860  CALC    APT←MPT
1861  EXIT    2               $$ FROM UNARY
1862  *
1863  *
1864  *-------------------
```

```
1905  * ---------------------------------------------------------
1906  *
                                                                                                                    BLD PAREN
                                                                                                                    BUILD
1908  UNIT    BUILD           $$ LEVEL 2 (ADD,SUB,MUL,DIV,POW)
1909  *
1910  *       CONSTRUCT INFIX FROM FRAGMENTS AT APT AND BPT
1911  *
1912  *       APT--- →TO SECOND OPERAND
1913  *              SET TO RESULT
1914  *       BPT--- →TO FIRST OPERAND
1915  *              DESTROYED
1916  *       OPR--- TYPE OF PRINCIPAL OPERATOR
1917  *
1918  *---    BOTH EVALUATABLE+/
1919  JOIN    PRIN(APT)<TYPE0$OR$PRIN(APT)>TYPEV,BUILD1,X
1920  JOIN    PRIN(BPT)<TYPE0$OR$PRIN(BPT)>TYPEV,BUILD1,X $$ TO INFIX
1921  *---    YES. EVALUATE AND SET AS RESULT.
1922  CALCS   OPR-TYPEADD-1,VAL↑V(TABLE↑INFO(APT))↑V(TABLE↑INFO(BPT))
1923          V(TABLE↑INFO(BPT))=V(TABLE↑INFO(APT)),
1924          V(TABLE↑INFO(APT))≠V(TABLE↑INFO(BPT)),
1925          V(TABLE↑INFO(BPT))≡V(TABLE↑INFO(APT)),
1926          V(TABLE↑INFO(BPT))**V(TABLE↑INFO(APT))
1927  CALC    MPT↑APT
1928          APT↑LPT↑BPT
1929  JOIN    DMOVEDU $$ TO BUMP SECOND FRAGMENT
1930  JOIN    LOOKC $$ TO FILL FIRST FRAGMENT ANEW
1931  EXIT    2               $$ FROM ADD SUB MUL DIV POW
1932  *
1933  *
1934  *
                                                                                                                    BUILD1
1935  UNIT    BUILD1          $$ LEVEL 3 (ADD,SUB,MUL,DIV,POW)
1936  *       CONSTRUCT NEW INFIX
1937  CALC    MPT↑APT
1938  CALCS   PRIN(APT)-TYPEADD,MSI↑-1,0,1,2,3,4,5 $$ SET TYPE APT
1939  CALCS   (OPR-TYPESUB)≠7↑MSI↑MSI↑0,-1,-1,0,0,0,0 $$ SUB/ALL
1940          0,-1,-1,0,-1,0,0 $$ MUL/ALL
1941          0,-1,-1,0,-1,0,0 $$ DIV/ALL
1942          0,-1,-1,-1,-1,-1,0 $$ POW/ALL
1943  JOIN    MSI,APAREN,X $$ ADD PAREN IF NECESSARY
1944  CALC    APT↑BPT $$ FOR USE OF APAREN
1945  CALCS   PRIN(APT)-TYPEADD,MSI↑-1,0,1,2,3,4,5 $$ SET TYPE APT
1946  CALCS   (OPR-TYPEMUL)≠7↑MSI↑MSI↑0
1947          -1,-1,0,-1,0,0 $$ MUL/ALL
1948          0↑-1↑-1↑0↑0↑0↑0↑0↑-1↑-1↑-1↑-1↑0
1949  JOIN    MSI,APAREN,X $$ ADD PAREN IF NECESSARY
1950  CALC    MSI↑1
1951  JOIN    DMOVEUP $$ TO OPEN UP ONE LOCATION
1952  CALC    N(MSI)↑ITYPE(OPR) $$ FILL TYPE FIELD
1953          APT↑MPT $$ UPDATE RESULT POINTER
1954          N(APT)↑N(APT)$MASK$ZPRIN$UNION$IPRIN(OPR)
1955  EXIT    1(APT)↑N(APT)$MASK$ZLENG$UNION$ILENG(LENG(BPT)+1+LENG(APT))
```

```
1956  * ----------------------------------------------------------
1957  *                                                                                  APAREN
1958  UNIT    APAREN
1961  *
1962  *       ADD PARENTHESES TO A FRAGMENT
1963  *
1964  *       APT--- ¬↑D FRAGMENT
1965  *
1966  CALC    N(APT)↠N(APT)$MASK$ZRPAR$UNION$IRPAR(RPAR(APT)+1)
1967          MO7↠APT-LENG(APT)+1
1968          N(MO7)↠N(MO7)$MASK$ZLPAR$UNION$ILPAR(LPAR(MO7)+1)
1969  *
1970  * ----------------------------------------------------------
1971  *                                                                                  SPAREN
1972  UNIT    SPAREN
1973  *
1974  *       REMOVE PARENTHESES FROM A FUNCTION-TYPE FRAGMENT
1975  *
1976  *       APT--- ¬↑D FRAGMENT
1977  *              PARENTHESES ARE REMOVED FROM N(APT) AND
1978  *                                   N(APT-LENG(APT)+2)
1979  *
1980  CALC    N(APT)↠N(APT)$MASK$ZRPAR$UNION$IRPAR(RPAR(APT)-1)
1981          MO7↠APT-LENG(APT)+2
1982          N(MO7)↠N(MO7)$MASK$ZLPAR$UNION$ILPAR(LPAR(MO7)-1)
1983  *
1984  * ----------------------------------------------------------
1985  *
  --------------------------------------------------------------------------------------------------------------------------------- INSRT LOOK
                                                                                     INSERT
1987  UNIT    INSERT          $$ WILL RESET TDERIV
1988  *
1989  *       INSERT A FRAGMENT FROM THE FORMEL INTO DERIV
1990  *
1991  *       INS--- -1↑: PUSH ON TOP, 0↑: PUSH BELOW TOP
1992  *       FROM--- -1↑: TAKE FROM TOP, 0↑: TAKE BELOW TOP
1993  *
1994  CALCS   FROM,MSI↠LENG(TFORMEL),LENG(TFORMEL-LENG(TFORMEL))
1995  CALC    MPT↠TDERIV
1996  JOIN    INS,X,DMOVEUP $$ OPEN SPACE IF BELOW TOP
1997  CALCS   INS,MSI↠(TDERIV↠TDERIV+MSI),MSI
1998  *ASSERT MSI↠TOP OF OPEN AREA
1999  *ASSERT TDERIV↠TOP OF DERIVATIVE
2000  INHIBIT ERASE
2001  JUMP    TDERIV>END,OVERFORM,X
2002  CALCS   FROM,MPT↠TFORMEL,TFORMEL-LENG(TFORMEL)
2003  *ASSERT MPT↠TOP OF AREA TO BE TAKEN
2004  CALC    MO1↠LENG(MPT) $$ NUMBER OF NODES TO BE TRANSPORTED
2005  1LOOP   N(MSI)↠N(MPT) $$ COPY
2006          MSI↠MSI-1
2007          MPT↠MPT-1
```

```
2008          BRANCH (MQ1=MQ1-1)>0,1LOOP,X $$ AGAIN
2009    *
2010    * ----------------------------------------------
2011    *                                                                    NODE
2012    UNIT      NODE            $$ WILL RESET TDERIV
2013    *
2014    *         INSERT A VALUE INTO THE DERIVATIVE
2015    *
2016    *         INS--- -1↑;PUSH ON TOP, 0↑; PUSH BELOW TOP
2017    *         VAL--- VALUE TO BE PUSHED
2018    *
2019    CALC      MSI=1
2020              MPT=TDERIV
2021    JOIN      INS,X,OMOVEUP $$ TO OPEN THE SPACE
2022    CALCS     INS,LPT=(TDERIV=TDERIV+1),MSI
2023    *ASSERT LPT=RECFIVING NODE
2024    INHIBIT ERASE
2025    JUMP      TDERIV>END,OVERFORM,X $$ BLEW AREA
2026    JOIN      LOOKC           $$ TO FILL NEW NODE
2027    *
2028    * ----------------------------------------------
2029    *                                                                    LOOKC
2030    UNIT      LOOKC
2031    *
2032    *         FIND OR ENTER A VALUE INTO THE TABLE
2033    *
2034    *         LPT--- =↑D FRAGMENT CONSISTING OF THAT CONSTANT
2035    *         VAL--- VALUE TO BE RECORDED
2036    *         WHE--- LINE =$ OF DISPLAY LINE
2037    *
2038    CALC      LJUNK=TABLE  $$ POINT TO BOTTOM OF TABLE
2039    1LOOP     BRANCH V(LJUNK)=VAL,2REC,X $$ SEARCH ON UP
2040              BRANCH (LJUNK=LJUNK+1)=<CPT,1LOOP,X
2041              CPT=LJUNK=CPT+1  $$ NOT FOUND, ENTER ANEW
2042              BRANCH CPT=>START,Q,X
2043              V(CPT)=VAL
2044    2REC      N(LPT)=IINFO(LJUNK-TABLE)$UNION$IWHER(WHE)$UNION$ILENG(1)
2045    INHIBIT ERASE
2046    JUMP      CPT=>START,OVERSYM,X $$ BLEW THE SYMBOLTABLE
2047    CALCS     LJUNK-TABLE-1,LJUNK=TYPE0,TYPE1,TYPEV $$ SET TYPE
2048    CALC      N(LPT)=N(LPT)$UNION$ITYPE(LJUNK)$UNION$IPRIN(LJUNK)
2049    *
2050    *
2051    *                                                                    OVERSYM
2052    UNIT      OVERSYM
2053    *
2054    *         TERMINAL↑! BLEW SYMBOL TABLE
2055    *
2056    MODE      REWRITE
2057    AT        1001
2058    ERASE     9
2059    WRITE     *S,X,S,T,A,T,E,M, ,A,R,E,A, ,I,N, ,O,V,E,R,F,L,O,W,S      CONSTANT TABLE OVERFLOWS
```

```
2060   ERASE   12
2062   JUMP    ERROR
2063   *
2064   * ----------
2065   *
```

```
                                                                                                                    MOVE STOP*
                                                                                                                    DMOVEUP
2067   UNIT    DMOVEUP         $$ WILL MODIFY TDERIV OR TFORMEL
2068   *
2069   *       MOVE UP IN FORMULA STACKS
2070   *
2071   *       MPT--- ¬↑D LOWEST FRAGMENT TO BE MOVED
2072   *              IS RESET TO NEW LOCATION OF FRAGMENT
2073   *       MSI--- ¬$ WORDS TO BE OPENED
2074   *              IS SET ¬↑D TOP OPEN AREA
2075   *
2076   *       ASSUMING↑1
2077   *       IF ITEM>MPT THEN MSI¬<(ITEM-TFORMEL)
2078   *
2079   JOIN    MPT>ITEM,MOVEUP1,MOVEUP2 $$ INITIALIZE
2080   *
2081   *       MOVE ONE FRAGMENT UP, DECIDE, AND CONTINUE
2082   *
2083   *       M01--- ¬↑D FRAGMENT BEFORE MOVE
2084   *              SET TO TOP AFTER MOVE
2085   *       MSI--- ¬$ OF ELEMENTS TO BE MOVED UP
2086   *       M02--- SET TO ¬↑D TOP OF NEXT FRAGMENT
2087   *       M03--- SET TO ¬↑D TOP OF OPEN AREA
2088   *
2089   CALC    M02↠M01-LENG(M01)
2090           M03↠M01
2091           M01↠M01+MSI
2092   1LOOP   N(M03+MSI)↠N(M03)
2093   CALC    BREAK$$RT
2094           BRANCH (M03↠M03-1)>M02,1LOOP,X
2095           BRANCH MPT=M01-MSI,3FIN,X
2096           M01↠M02+MSI
2097           M02↠M02-LENG(M02)
2098           BRANCH 1LOOP
2099   3FIN    MPT↠M01
2100           MSI↠M03+MSI
2101   *
2102   *
2103   *
```

```
                                                                                                                    MOVEUP1
2104   UNIT    MOVEUP1
2105   *       INITIALIZE THE MOVE IN THE DERIVATIVE
2106   INHIBIT ERASE
2107   JUMP    TDERIV+MSI>END,OVERFORM,X $$ BLEW SIZE
2108   CALC    M01↠TDERIV
2109           TDERIV↠TDERIV+MSI
2110   *
2111   *
```

```
2112  *                                                                                    MOVEUP2
2113  UNIT    MOVEUP2
2114  *       INITIALIZE A MOVE WITHIN SLACK
2115  INHIBIT ERASE
2116  JUMP    TFORMEL↑MSI>ITEM,OVERSLAC↑X $$ SYSTEM FAILURE
2117  CALC    MO1↠TFORMEL
2118          TFORMEL↠TFORMEL↑MSI
2119  *
2120  *
2121  *                                                                                    OVERFORM
2122  UNIT    OVERFORM
2123  *       SYSTEM FAILURE↑; OVERFLOWING FORMULA
2124  MODE    REWRITE
2125  AT      1001
2126  ERASE   9
2127  WRITE   ↑S↑Y↑S↑T↑E↑M ↑F↑A↑I↑L↑U↑R↑E --- DERIVATIVE EXCEEDS AREA
2128  ERASE   13
2129  PAUSE
2130  JUMP    ERROR
2131  *
2132  *
2133  *                                                                                    OVERSLAC
2134  UNIT    OVERSLAC
2135  *       SYSTEM FAILURE↑; EXCEEDING SLACK
2136  MODE    REWRITE
2137  AT      1001
2138  ERASE   9
2139  WRITE   ↑S↑Y↑S↑T↑E↑M ↑F↑A↑I↑L↑U↑R↑E --- UNEXPECTED EXCEEDING OF SLACK
2140  ERASE   7
2141  PAUSE
2142  JUMP    ERROR
2143  *
2144  * ---------------------------------------------------------
2145  *                                                                                    DMOVEDO
2146  UNIT    DMOVEDO         $$ WILL MODIFY TDERIV OR TFORMEL
2147  *
2148  *       DOWNWARD MOVE IN FORMULAS
2149  *
2150  *       MPT--- →↑D FRAGMENT TO BE BUMPED
2151  *              IS SET TO LOCATION OF OVERWRITING FRAGMENT
2152  *       MSI--- SET TO →$ OF WORDS BUMPED
2153  *
2154  CALC    MSI↠LENG(MPT) $$ GET LENGTH
2155          BRANCH MPT>ITEM↑2ABO↑X
2156          MO1↠TFORMEL↠TFORMEL-MSI
2157          BRANCH 2DIR
2158  2ABO    MO1↠TDERIV↠TDERIV-MSI $$ TOP AFTER MOVE
2159  2DIR    MPT↠1↑(MO2↠MPT-MSI) $$ MO2 UNDISTURBED TOP
2160  3LOOP   N(MPT)↠N(MPT↑MSI) $$ MOVE
2161  CALC    BREAK$$RT
2162          BRANCH (MPT↠MPT↠1)=<MO1↑3LOOP↑X
```

```
2164   3LOOP    N(MPT)=N(MPT+MSI)  ;; MOVE
2165   CALC     BREAK$RT
2165   *LOOP    BRANCH (MPT=M01)-M02;0,V ;; TOO WAS ALWAYS
2165   CALC     BREAK$RT
2166            MPT=MPT-LENG(MPT)
2167            BRANCH 4LOOP
2168   *
2169   * ---------------------------------------
2170   *
2171   STOP*
```

UNIT	BLOCK	UNIT LOCATION	REFERENCES TO UNIT						
ADD	ADD MUL	1701	1306	1318					
ADD1	ADD MUL	1720	1711	1748	1778	1795	1812		
ADD2	ADD MUL	1729	1713	1780					
ANSYES	OPTIONS3	1155	1144						
APAREN	BLD PAREN	1960	1869	1870	1943	1949			
BUILD	BLD PAREN	1908	1716	1753	1783	1798	1817		
BUILD1	BLD PAREN	1935	1919	1920					
CALC2	TITLE	22							
DATELINE	TITLE	55	37	361	626	941	942	1097	1143
			1171	1608	1649				
DEBUG	DBG INDEX	233	259	266	267				
DEBUGC	DBG INDEX	327	306						
DEBUGV	DBG INDEX	325	306	306	306				
DEBUG1	DBG INDEX	263	237						
DEBUG2	DBG INDEX	302	298						
DERIVE	DERIVE1	1165	1096	1150	1213	1365			
DERUNA	DERIVE2	1298	1200						
DERUNA1	DERIVE2	1310	1303						
DERVAR	DERIVE2	1281	1200						
DISPLAY1	DERIVE1	1229	1199	1249					
DISPLAY2	DERIVE1	1248	1199	1261					
DISPLAY4	DERIVE2	1260	1199						
DIV	DIV POW	1785	1306	1318					
DMOVEDO	MOVE STOP*	2146	1724	1732	1761	1844	1859	1929	
DMOVEUP	MOVE STOP*	2067	1873	1951	1996	2041			
ERROR	DEFINE1	75	235	268	567	578	589	729	913
			1225	1657	2062	2130	2142		
ERROR1	OPTIONS1	941	950						
HELPA	QUESTION3	1591	1561	1573	1584				
HELP1	QUESTION1	1343	1324						
HELP2	QUESTION2	1416	1371						
HELP4	QUESTION2	1488	1428						
HELP41	QUESTION2	1509	1421						
INDEX	DBG INDEX	207	210						
INFIX	INFIX	837	1204	1237	1254	1267	1277	1293	1615
INFIX1	INFIX	852	847						
INFIX2	INFIX	860	667	854					
INFIX3	INFIX	866	695	855					
INFIX4	INFIX	873	648	855	855	855			
INFIX5	INFIX	880	855						
INFIX6	INFIX	891	747	856					
INFIX7	INFIX	897	874						
INHIBIT	DERIVE1	1218	1212						
INSERT	INSRT LOOK	1987	1318						
JUDGE*S	QUESTION3	1525	1408	1475					
LOOKC	INSRT LOOK	2030	644	1031	1289	1847	1902	1930	2026
MOVEUP1	MOVE STOP*	2104	2079						
MOVEUP2	MOVE STOP*	2113	2079						
MUL	ADD MUL	1768	1306	1318					
MYSELF	TITLE	9							
NEXTITEM	QUESTION1	1359	1332	1580	1600				
NOVE	INSRT LOOK	2012	1318						
NONEREAD	READIN3	582	383						
OPTB	OPTIONS1	953	960	1657					

OPTB1	OPTIONS1	960	968						
OPTB2	OPTIONS1	966	963	964					
OPTB3	OPTIONS1	979	965	1673					
OPTC	OPTIONS3	1076	950						
OPTC1	OPTIONS3	1139	1135	1158					
OPTD	OPTIONS1	995	950						
OPTD1	OPTIONS2	1024	1016						
OPTD2	OPTIONS2	1049	1021						
OPTD3	OPTIONS2	1056	1050						
OPTD4	OPTIONS2	1062	1051						
OPTIONS	OPTIONS1	918	631	976	980	998	1017	1022	1078
			1142						
OPT*C	OPTIONS3	1068	950						
OUTPUT	OUTPUT	790	926	1102	1175	1238	1255	1268	1278
			1599	1617	1626				
OUTPUT1	OUTPUT	803	799						
OUTPUT2	OUTPUT	812	804						
OUTPUT3	OUTPUT	820	805						
OUTPUT4	OUTPUT	827	806						
OVERAREA	READIN3	560	383	551	621	1698			
OVERCORE	INFIX	903	665	717	763	862	869	876	886
			893	899					
OVERFORM	MOVE STOP*	2122	1288	2001	2025	2107			
OVERSLAC	MOVE STOP*	2134	2116						
OVERSTAC	POSTFIX2	722	661	715	764				
OVERSYM	INSRT LOOK	2052	2046						
PFSTACK	OUTPUT	767	628	744	754				
PF11	POSTFIX1	641	636						
PF111	POSTFIX2	670	647						
PF112	POSTFIX2	678	647						
PF12	POSTFIX1	653	636						
PF13	POSTFIX2	685	637						
PF14	POSTFIX2	700	637						
PF15	POSTFIX2	707	637						
PF21	POSTFIX2	739	735						
PF22	POSTFIX2	752	735	735	735	735	735		
POSTFIX	POSTFIX1	591	552						
POSTFIX0	POSTFIX1	617	613						
POSTFIX1	POSTFIX1	635	619						
POSTFIX2	POSTFIX2	733	619						
POSTFIX3	POSTFIX1	625	620						
POW	DIV POW	1802	1306	1318					
POW1	DIV POW	1821	1814						
QUESTA	QUESTION3	1543	1407	1472	1519				
QUESTA1	QUESTION3	1557	1576	1587					
QUEST1	QUESTION1	1321	1214						
QUEST2	QUESTION1	1368	1214						
QUEST4	QUESTION2	1425	1214						
RANDOM	QUESTION2	1522	1391	1447	1538	1564			
READIN	READIN1	339	51	205	337	950			
READIN2	READIN3	548	388	408	418	462	486	512	1699
READ2	READIN1	392	555						
READ21	READIN1	412	553						
READ3	READIN2	420	556						
READ31	READIN2	466	554						

UNIT	BLOCK	UNIT LOCATION	REFERENCES TO UNIT						
READ32	READIN3	488	554						
READ4	READIN3	516	556						
RETAIN	RETAIN	1602	976	980	1213	1365	1684		
RETAINC	RETAIN	1661	1657						
RETAINC1	RETAIN	1669	1676						
RETAINC2	RETAIN	1674	1671	1672					
RETAIND	RETAIN	1688	1657	1657					
RETAIN1	RETAIN	1648	1657						
SPAREN	BLD PAREN	1972	1853	1854					
SUB	ADD MUL	1738	1306	1318					
SUB1	ADD MUL	1757	1750						
SYNTAX	READIN3	571	550	627	636	637	735	735	735
			735	741					
TOOMUCH	READIN1	331	216	218	229	231			
UNARY	UNARY	1833	1306	1318	1318	1763			
UNARY1	UNARY	1851	1842	1844	1845				
UNARY2	UNARY	1865	1843	1846					
UNARY3	UNARY	1882	1868						

A	STUDENT	77								
ABS	MINE	671	830	1843						
ACOS	STUDENT	88								
APT	MINE	847	847	847	1201	1233	1251	1252	1265	1274
		1275	1290	1305	1312	1614	1709	1709	1711	1722
		1723	1733	1746	1746	1748	1762	1776	1776	1778
		1780	1793	1793	1795	1810	1810	1812	1814	1825
		1842	1843	1844	1854	1854	1855	1855	1855	1855
		1856	1856	1856	1857	1857	1860	1868	1870	1871
		1874	1875	1875	1876	1877	1877	1884	1901	1919
		1919	1922	1923	1924	1925	1926	1927	1928	1937
		1938	1944	1945	1953	1954	1954	1955	1955	1955
		1966	1966	1966	1967	1967	1980	1980	1980	1981
		1981								
ARCTAN	STUDENT	87	88	89						
ARCTAN	MINE	1889	1891	1893						
ASIN	STUDENT	87								
ATAN	STUDENT	89								
B	STUDENT	77								
BPT	MINE	646	690	691	691	847	854	855	856	868
		874	875	881	883	1709	1713	1723	1731	1746
		1750	1759	1776	1778	1780	1793	1795	1810	1812
		1812	1823	1920	1920	1922	1923	1924	1925	1926
		1928	1944	1955						
C	STUDENT	77								
C	MINE	1535								
COPT	MINE	1198	1199	1200	1214	1233	1251	1300	1300	1406
		1406	1420	1420	1470	1470	1492	1492	1537	1537
COS	STUDENT	85	85	86						
COS	MINE	1890	1892	1894	1895					
COSH	STUDENT	90								
COT	STUDENT	85								
CPT	MINE	359	1697	2041	2041	2042	2043	2046		
CSC	STUDENT	86								
D	STUDENT	77								
DATE1	DATES	60	63	65						
DATE2	DATES	60	64	69						
DERIV	MINE	287	629	968	1016	1028	1030	1038	1042	1045
		1087	1095	1213	1365	1665				
DOPT	MINE	1181	1183	1183	1184	1187	1190	1191	1194	1195
		1198	1205	1211	1331	1333	1335	1348	1351	1354
		1356	1384	1442	1473	1474	1496	1512	1515	1544
		1548	1562	1565	1570	1574	1581	1585	1593	1595
DVAR	MINE	1085	1108	1114	1242	1272	1374	1382	1431	1440
		1621								
E	STUDENT	77								
END	MINE	383	385	552	554	613	1288	1694	1698	2001
		2025	2107							
ERROR	MINE	216	229							
EXP	STUDENT	89	89	90	90	91	91	91	91	92
EXP	MINE	1887	1897	1897	1898	1898	1899			
F	STUDENT	77								
FORLOC	MINE	607	661	715	764	924	1100	1174	1613	1692
		1692								
FORMAX	MINE	609	1612	1692						

SYMBOL	DEFINE SET	REFERENCES TO SYMBOL								
FORMLEN	MINE	673	1034							
FORMULA	MINE	969	1052	1057	1063					
FORSIZE	MINE	630	923	969	1050	1099	1173	1616		
FROM	MINE	1315	1994	2002						
G	STUDENT	77								
H	STUDENT	77								
HELP	MINE	1573	1584							
HELPA	REQUEST	12	17							
HELPA	MINE	1598	1598							
HELPS	REQUEST	13	17							
HELPS	MINE	1345	1345	1513	1513					
I	STUDENT	78								
IINFO	MINE	95	690	2044						
ILENG	MINE	98	691	780	1856	1858	1875	1955	2044	
ILPAR	MINE	99	1968	1982						
INFO	MINE	109	326	328	868	874	875	964	987	989
		1020	1027	1093	1672	1884	1922	1922	1923	1923
		1924	1924	1925	1925	1926	1946			
INLIST	MINE	666	716	762	861	867	868	875	887	892
		898	1052	1063						
INS	MINE	1311	1312	1996	1997	2021	2022			
INT	MINE	671	774	774	830	1177	1844	1845		
IPRIN	MINE	97	691	780	1093	1855	1876	1877	1954	2048
IRPAR	MINE	100	1966	1980						
ITEM	MINE	291	1089	1176	1178	1207	1207	1213	1233	1283
		1304	1365	2079	2116	2155				
ITYPE	MINE	96	690	780	1094	1876	1952	2048		
IWHER	MINE	101	690	781	1041	1043	2044			
J	STUDENT	78								
JCOUNT	MINE	383	383	385	1400	1464	1694	1698		
K	STUDENT	78								
KEY	MINE	1005	1085	1573	1584					
L	STUDENT	78								
LENG	MINE	112	318	847	1186	1189	1193	1233	1251	1301
		1312	1709	1746	1776	1793	1810	1854	1855	1856
		1857	1955	1955	1967	1981	1994	1994	1994	2002
		2004	2089	2097	2154	2164	2166			
LJUNK	MINE	2038	2039	2040	2040	2041	2044	2047	2047	2048
		2048								
LN	MINE	671	671	830	830	1886				
LOPR	MINE	776	778	780	780					
LPAR	MINE	113	320	854	1968	1982				
LPARCT	MINE	605	627	660	660	741	785	785		
LPT	MINE	604	621	629	646	646	690	693	693	780
		781	781	782	782	783	1015	1026	1027	1028
		1028	1030	1031	1033	1035	1037	1037	1043	1045
		1284	1825	1884	1885	1886	1887	1888	1889	1889
		1890	1891	1891	1892	1892	1893	1894	1894	1895
		1896	1897	1897	1898	1898	1899	1900	1901	1928
		2022	2044	2048	2048					
M	STUDENT	78								
MALLOT	MINE	216	229	334						
MOI	MINE	200	201	202	202	203	203	203	204	204
		313	316	317	318	319	441	442	443	444

		403	407	415	416	416	422	423	424	425
		426	428	429	430	431	432	433	434	435
		438	438	439	439	441	445	445	459	460
		460	468	469	470	471	478	478	480	483
		484	484	490	491	492	493	494	503	503
		504	504	506	509	510	510	518	519	520
		521	522	523	524	525	527	527	542	543
		543	550	551	552	553	554	554	555	556
		606	619	627	674	681	694	759	798	807
		807	808	814	815	822	829	832	832	854
		856	948	950	1005	1018	1018	1027	1058	1069
		1070	1070	1071	1071	1090	1091	1092	1092	1093
		1093	1093	1094	1094	1094	1095	1095	1176	1177
		1177	1177	1179	1205	1208	1209	1210	1210	1211
		1354	1355	1355	1356	1512	1514	1514	1515	1643
		1644	1645	1645	1646	1646	1646	1647	1647	1655
		1657	1692	1692	1692	1694	2004	2008	2008	2089
		2089	2090	2091	2091	2095	2096	2099	2108	2117
		2156	2158	2163						
MU1LOC	MINE	671	672							
MU2	MINE	387	394	395	395	401	401	404	404	405
		405	414	415	417	417	426	436	440	440
		442	444	447	449	451	453	455	457	461
		461	471	473	475	477	479	479	481	485
		485	494	496	498	500	502	505	505	507
		511	511	526	529	531	533	535	537	539
		539	540	544	544	551	620	806	813	815
		816	881	887	1013	1020	1031	1032	1033	1043
		1051	1052	1177	1179	1179	1180	1208	1210	1212
		1213	1313	1318	1696	2089	2094	2096	2097	2097
		2097	2159	2163	2164					
MU2LOC	MINE	830	831	1020						
MU3	MINE	396	398	399	402	402	402	404	404	613
		821	822	823	828	829	830	831	883	886
		887	887	1014	1017	1021	1021	1025	1025	1029
		1029	1209	1210	2090	2092	2092	2094	2094	2100
MU4	MINE	662	666	667	799	804	804	805	807	808
		822	822	823	1034	1036	1041	1057	1058	
MU5	MINE	663	664	664	665	666	666	804	805	807
		814	815	822	823	829	1037	1038	1039	1041
		1041	1042	1042						
MU6	MINE	1016								
MU7	MINE	1967	1968	1968	1968	1981	1982	1982	1982	
MPT	MINE	611	620	621	636	637	643	645	645	655
		657	657	688	690	692	692	702	702	712
		712	734	743	743	757	757	783	1722	1731
		1733	1753	1762	1823	1825	1857	1858	1860	1871
		1874	1927	1937	1953	1995	2002	2004	2005	2007
		2007	2020	2079	2095	2099	2154	2155	2159	2159
		2160	2160	2162	2162	2163	2164	2164	2166	2166
		2166								
MSI	MINE	1872	1876	1938	1939	1939	1943	1945	1946	1946
		1949	1950	1952	1994	1997	1997	1997	2005	2006
		2006	2019	2022	2091	2092	2095	2096	2100	2100

SYMBOL	DEFINE SET	REFERENCES TO SYMBOL								
		2107	2109	2116	2118	2154	2156	2158	2159	2160
MUSE	MINE	216	229	334						
N	REQUEST	14	15	16						
N	STUDENT	78								
N	MINE	109	110	111	112	113	114	115	201	202
		202	203	203	203	385	394	397	401	414
		415	422	423	424	425	426	426	428	429
		430	431	432	433	434	435	436	441	442
		444	447	449	451	453	455	457	468	469
		470	471	471	473	475	477	480	481	490
		491	492	493	494	494	496	498	500	502
		506	507	518	519	520	561	522	523	524
		525	526	529	531	533	535	537	539	540
		553	554	555	556	636	637	655	655	658
		662	663	666	688	690	690	691	691	710
		713	716	734	755	756	762	773	774	775
		776	776	777	778	779	780	781	781	781
		785	786	815	822	829	861	867	868	875
		887	892	898	955	964	1033	1041	1041	1043
		1070	1070	1072	1072	1092	1092	1093	1093	1094
		1094	1096	1101	1101	1205	1211	1283	1283	1354
		1356	1512	1515	1644	1645	1645	1646	1646	1646
		1663	1672	1692	1692	1694	1855	1855	1856	1856
		1858	1875	1875	1876	1877	1877	1952	1954	1954
		1955	1955	1966	1966	1968	1968	1980	1980	1982
		1982	2005	2005	2044	2048	2048	2092	2092	2160
		2160								
NC	MINE	1198	1199	1200	1214	1233	1251	1300	1303	1311
		1313	1314	1314	1315	1316	1317	1406	1406	1420
		1420	1470	1470	1492	1492	1537	1537		
NC551	MINE	256	258	328	329					
NEXT	MINE	1691								
N1	MINE	273	275	277						
N146	REQUEST	12								
N147	REQUEST	13								
N8	DATES	60								
N9	DATES	60								
O	STUDENT	78								
OPR	MINE	628	734	735	742	755	760	761	762	762
		763	772	774	775	1178	1181	1181	1285	1304
		1306	1314	1314	1715	1752	1760	1782	1797	1816
		1842	1843	1843	1843	1844	1844	1844	1845	1853
		1854	1869	1870	1876	1877	1885	1922	1939	1946
		1952	1954							
P	STUDENT	78								
PAR1	MINE	954	955	956	956	957	963	964	968	968
		969	987	989	1300	1303	1311	1313	1314	1314
		1315	1316	1317	1317	1317	1317	1408	1410	1473
		1475	1475	1477	1477	1535	1662	1663	1664	1664
		1665	1671	1672	1676	1676	1677			
PAR2	MINE	925	970	976	980	981	986	988	990	1303
		1391	1447	1523	1538	1564	1578	1579	1625	
		1713	1748	1750	1778	1779	1780	1784	1787	1789

Q	STUDENT	79								
R	STUDENT	79								
REQUEST	REQUEST	11	14	15	16					
REQUEST	MINE	200	1069	1072	1072	1096	1101	1101	1205	1211
		1354	1356	1512	1515	1643				
RPAR	MINE	114	322	856	1966	1980				
S	STUDENT	79								
SEC	STUDENT	86								
SHIFT	MINE	553	806	816						
SIN	STUDENT	85	85	86						
SIN	MINE	1888	1892	1894	1896					
SINH	STUDENT	89								
SQRT	STUDENT	87	88							
SQRT	MINE	1889	1891	1900						
STACK	MINE	603	771	786						
START	MINE	298	303	305	307	312	317	319	321	323
		383	387	604	611	613	957	1015	1016	1088
		1089	1090	1696	1698	2042	2046			
STOP	MINE	603	655	658	659	659	661	662	663	710
		713	714	714	715	755	756	758	758	764
		771	773	774	775	776	776	777	778	779
		781	784	784	785	786	786	1474	1475	1477
T	STUDENT	79								
TABLE	MINE	326	357	358	359	415	830	831	1697	1884
		1922	1922	1923	1923	1924	1924	1925	1925	1926
		1926	2038	2044	2047					
TAN	STUDENT	85								
TANH	STUDENT	91								
TDERIV	MINE	298	1087	1185	1186	1186	1188	1189	1189	1192
		1193	1193	1201	1251	1251	1251	1274	1284	1284
		1288	1290	1301	1301	1302	1312	1312	1995	1997
		1997	2001	2020	2022	2022	2025	2107	2108	2109
		2109	2158	2158						
THERMEL	MINE	283	1088	1233	1233	1233	1285	1283	1283	1286
		1305	1614	1994	1994	1994	2002	2002	2002	2116
		2117	2118	2118	2156	2156				
TYPE	MINE	110	306	308	311	855	881	883	963	1026
		1035	1091	1178	1304	1671				
TYPEACS	MINE	133								
TYPEADD	MINE	122	776	881	883	1306	1715	1922	1938	1945
TYPEASN	MINE	131								
TYPEATN	MINE	135	1843							
TYPEC	MINE	118	690	691						
TYPECOS	MINE	132								
TYPECOT	MINE	136								
TYPEDIV	MINE	125	1797							
TYPEEXP	MINE	129								
TYPELOG	MINE	128	778							
TYPEMIN	MINE	127	1314	1760	1842	1842	1843	1853	1854	1869
		1870	1885							
TYPEMUL	MINE	124	1782	1946						
TYPEPOW	MINE	126	1816							
TYPESEC	MINE	137								
TYPESIN	MINE	130								

SYMBOL	DEFINE SET	REFERENCES TO SYMBOL								
TYPESUB	MINE	123	1752	1939						
TYPETAN	MINE	134								
TYPEV	MINE	121	1035	1854	1870	1919	1920	2047		
TYPEX	MINE	117	855	1093						
TYPFO	MINE	119	1301	1302	1711	1713	1748	1750	1778	1780
		1795	1812	1814	1919	1920	2047			
TYPE1	MINE	120	1778	1780	1795	1812	1812	2047		
TYP*D*U	MINE	1182	1252	1301						
TYP*D*V	MINE	1182	1275	1302						
U	STUDENT	79								
V	STUDENT	79								
V	MINE	326	357	358	359	395	401	401	404	404
		415	415	643	671	672	830	830	831	831
		989	1523	1885	1886	1887	1888	1889	1889	1890
		1891	1891	1892	1892	1893	1894	1894	1895	1896
		1897	1897	1898	1898	1899	1900	1922	1922	1923
		1923	1924	1924	1925	1925	1926	1926	2039	2043
VAL	MINE	643	647	647	647	671	672	679	969	973
		1009	1285	1316	1328	1406	1407	1470	1472	1537
		1539	1579	1580	1677	1681	1846	1885	1899	1899
		1899	1899	1922	2039	2043				
VALUE	MINE	1185	1186	1188	1189	1192	1193			
VC551	MINE	244	252	254						
V1	STUDENT	77	84							
V1	MINE	279								
V10	STUDENT	78								
V11	STUDENT	78								
V12	STUDENT	78								
V13	STUDENT	78								
V14	STUDENT	78								
V15	STUDENT	78								
V16	STUDENT	78								
V17	STUDENT	79								
V18	STUDENT	79								
V19	STUDENT	79								
V2	STUDENT	77	84							
V20	STUDENT	79								
V21	STUDENT	79								
V22	STUDENT	79								
V23	STUDENT	79								
V24	STUDENT	79								
V25	STUDENT	80								
V26	STUDENT	80								
V3	STUDENT	77								
V4	STUDENT	77	84							
V5	STUDENT	77	84							
V6	STUDENT	77								
V60	STUDENT	81								
V61	STUDENT	81								
V62	STUDENT	81								
V63	STUDENT	81								
V64	STUDENT	81								
V65	STUDENT	81								

V68	STUDENT	82								
V69	STUDENT	82								
V7	STUDENT	77								
V70	STUDENT	82								
V71	STUDENT	82								
V72	STUDENT	82								
V73	STUDENT	82								
V74	STUDENT	83								
V8	STUDENT	77								
V9	STUDENT	78								
W	STUDENT	79								
WHE	MINE	610	658	664	664	673	673	680	680	689
		689	690	710	711	711	746	746	756	761
		761	1031	1040	1043	1286	2044			
WHER	MINE	115	324	1031	1039	1176	1286			
WHERE	MINE	972	1081	1620	1680					
WORKFOR	MINE	1579	1677							
WORKLOC	MINE	1202	1235	1291	1597	1624	1692			
WORKMAX	MINE	1203	1236	1253	1266	1276	1292			
WORKSIZ	MINE	1206	1294	1579	1596	1623	1677			
X	STUDENT	79								
X	MINE	1314								
XPT	MINE	607	666	716	762	815	842	829	861	867
		868	875	887	892	898	924	1100	1174	1202
		1235	1291	1597	1613	1624				
XSI	MINE	609	665	717	763	799	808	814	848	862
		869	876	886	893	899	923	1099	1173	1203
		1206	1236	1253	1266	1276	1292	1294	1596	1612
		1616	1623							
XUS	MINE	608	630	665	665	666	716	717	717	762
		763	763	846	848	861	862	862	867	868
		869	869	875	876	876	886	886	887	892
		893	893	898	899	899	1019	1039	1040	1041
		1050	1050	1052	1052	1057	1063			
Y	STUDENT	80								
Z	STUDENT	80								
ZINFO	MINE	102								
ZLENG	MINE	105	1856	1955						
ZLPAR	MINE	106	1968	1982						
ZPRIN	MINE	104	1092	1855	1877	1954				
ZRPAR	MINE	107	1966	1980						
ZTYPE	MINE	103	1092							
ZWHER	MINE	108	1033	1041						
ZZ	STUDENT	85	85	85	85	85	85	86	86	86
		86	87	87	87	88	88	88	89	89
		89	89	89	90	90	90	91	91	91
		91	91							
ZZ	MINE	95	95	96	96	97	97	98	98	99
		99	100	100	101	101	109	109	110	110
		111	111	112	112	113	113	114	114	115
		115								
↑A	STUDENT	81								
↑B	STUDENT	81								
↑C	STUDENT	81								
↑D	STUDENT	81								

SYMBOL	DEFINE SET	REFERENCES TO SYMBOL							
↑DFORM	MINE	1400	1464	1536					
↑DPT	MINE	1405	1469	1536					
↑DSIZE	MINE	1400	1464	1536					
↑DVAL	MINE	1401	1407	1465	1472	1536	1539	1569	1580
↑D↑U	STUDENT	84							
↑D↑U	MINE	1535	1535						
↑D↑V	STUDENT	84							
↑D↑V	MINE	1535	1535						
↑E	STUDENT	81							
↑F	STUDENT	81							
↑G	STUDENT	81							
↑H	STUDENT	82							
↑I	STUDENT	82							
↑J	STUDENT	82							
↑K	STUDENT	82							
↑L	STUDENT	82							
↑M	STUDENT	82							
↑N	STUDENT	82							
↑O	STUDENT	83							
↑SPT	MINE	1405	1406	1469	1470	1537			
↑U	STUDENT	84							
↑V	STUDENT	84							
↑7CALCN	MINE	228	230						
↑7INDEX	MINE	217							
↑7NOTE↑	MINE	230							
↑7UTCAL	MINE	215	217						
¬A	STUDENT	92							
¬M	STUDENT	92							
¬P	STUDENT	88	92						
¬P	MINE	1891							

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-73-611	2.	3. Recipient's Accession No.
4. Title and Subtitle calc2 - A PLATO IV Lesson on Differentiation			5. Report Date December 1973 6.
7. Author(s) Axel T. Schreiner			8. Performing Organization Rept. No.
9. Performing Organization Name and Address University of Illinois at Urbana-Champaign Departments of Computer Science and Mathematics Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address University of Illinois at Urbana-Champaign Departments of Computer Science and Mathematics Urbana, Illinois 61801			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

calc2 is a lesson differentiation operating on the PLATO IV computer-aided instruction system. A symbolic differentiator guides the student through all steps of the differentiation of a student-chosen formula. The report describes the details of the table-driven symbolic differentiator.

17. Key Words and Document Analysis. 17a. Descriptors

calculus
computer-aided instruction
symbolic differentiation
derivative

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
RELEASE UNLIMITED	UNCLASSIFIED	84
	20. Security Class (This Page) UNCLASSIFIED	22. Price

JUL 16 1924

BOUND BY
HNM

UNIVERSITY OF ILLINOIS-URBANA
3 0112 064441527